Oracle9*i*AS Portal

Configuration Guide

Release 2 (9.0.2)

May 2002

Part No. A90852-02

ORACLE

Oracle9*i*AS Portal Configuration Guide, Release 2 (9.0.2)

Part No. A90852-02

Primary Author: Peter Lubbers

Contributors: Dave Mathews, Deanna Kitis, Paul Encarnacion, Todd Vender, Susan Highmoor, Frank Rovitto, Cheryl Smith, Mark Clark, Rajiv Chopra, Demetris Christou, Pushkar Kapasi, Bill. Lankenau, Eric Lee, Mark Loper, Sunil Marya, Rich Soule, Arun Arat Tharakkal, P.V. Dharan, Dawn Tyler, Chris van Es, Paul Spencer, Chung-Ho Chen, Eddy Chee, Ross Clewley, Senthil Arunagirinathan.

Contents

6 Advanced Oracle9*i*AS Portal Configuration

7 Configuring the Search Features in Oracle9*i*AS Portal

8 Troubleshooting Oracle9*i*AS Portal

A Oracle9*i* Application Server Configuration Files

B Oracle9*i*AS Portal Installation and Configuration Scripts

Index

Send Us Your Comments

Oracle9*i*AS Portal Configuration Guide, Release 2 (9.0.2)

Part No. A90852-02

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send us comments via electronic mail at `webdbdoc_us@oracle.com`. Alternatively, you can send comments to us via the Oracle9*i*AS Portal discussion group forum on the Oracle Technology Network at:

`http://otn.oracle.com`

If you would like a reply, please give your name, address, telephone number, and electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This guide provides information about configuring Oracle9*i*AS Portal.

Intended Audience

This guide is intended for users who are responsible for configuring and maintaining Oracle9*i*AS Portal.

> **Note:** An updated version of this manual may be available at:
>
> `http://portalcenter.oracle.com`

Structure

This guide is comprised of the following:

Chapter	Contents
Chapter 1, "Verifying Requirements"	Describes the Oracle9*i*AS Portal system requirements.
Chapter 2, "Post-installation Tasks"	Provides general post-installation information, including how to access Oracle9*i*AS Portal in your browser, and descriptions of the default Oracle9*i*AS Portal schemas, accounts, and groups created upon installation.

Chapter	Contents
Chapter 3, "Configuring Oracle9iAS Portal using OPCA"	Most of the Oracle9*i*AS Portal configuration will now be done with the use of the Oracle9*i*AS Portal Configuration Assistant. This section provides information about the various topologies that you can set up with Oracle9*i*AS Portal including stand-alone laptop, integrated server, and traditional three-tier. It also discusses installing and changing the language of Oracle9*i*AS Portal in your browser.
Chapter 4, "Using the PL/SQL HTTP Adapter"	Provides information about the PL/SQL HTTP Adapter and on how to set up the environment for it, and how to use it to share portlets with other Oracle9*i*AS Portal instances.
Chapter 5, "Deploying Web Portals"	Provides information on various deployment configurations.
Chapter 6, "Advanced Oracle9iAS Portal Configuration"	Provides instructions on how to perform more advanced Oracle9*i*AS Portal configurations, including middle Tier, proxy server, Oracle9*i*AS Web Cache and Oracle9*i*AS Single Sign-On Server configuration.
Chapter 7, "Configuring the Search Features in Oracle9iAS Portal"	Provides instructions on configuring Oracle Text to perform text searching in page groups created with Oracle9*i*AS Portal and information on how to set up and start using Oracle Ultra Search.
Chapter 8, "Troubleshooting Oracle9iAS Portal"	Provides solutions to problems you may encounter while installing or using Oracle9*i*AS Portal.
Appendix A, "Oracle9i Application Server Configuration Files"	Provides information about the configuration files which can affect the connection to and the behavior of the Oracle9*i* Application Server and its components in the middle tier as well as on other machines to which it is connecting.
Appendix B, "Oracle9iAS Portal Installation and Configuration Scripts"	Provides information about various scripts and OPCA modes that are used for customizing the configuration.

Conventions

The following notational conventions are used in this guide:

Convention	Meaning
boldface	Used for emphasis. Also used for button names, labels, links, and other user interface elements.
italics	Used to introduce new terms and book titles. Also used to represent a variable. Substitute an appropriate value for the italic text.
`courier`	Used to represent text you need to type. Also used for file names and directories.
CAPS	Used for environment variables, command line keywords, and built-ins and package names on an NT platform. The UNIX platform uses lower case.
ORACLE_HOME	Refers to the location of the Oracle9*i* Application Server installation files, including those for the Oracle9*i*AS Portal component.

Documentation

The Oracle9*i*AS Portal Online Help page group, is an online help system which provides detailed step-by-step instructions and reference information, as well as an introduction to Oracle9*i*AS Portal and troubleshooting information.

This guide refers you to various "topics" included in the Oracle9*i*AS Portal online help system.

You can find information about the Oracle9*i* Application Server in the Oracle9*i* Application Server documentation library, available at:

`http://otn.oracle.com`

Oracle9*i*AS Portal Publications

You can also refer to the following publications which are available from the Oracle Technology Network at:

`http://portalcenter.oracle.com`

Part Number	Title	Description
A96191-01	*Release Notes*	Describes last minute changes to the product or documentation.
A90098-01	*Building Advanced Portals*	Provides several cases that show you how to use Oracle9*i*AS Portal's advanced features.
N / A	*Oracle9iAS Portal Development Kit*	Provides detailed information about the Oracle9*i*AS Portal API set, as well as numerous examples that demonstrate API implementation.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at:

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Oracle Support Services

A wide range of information about Oracle products and services is available at:

`http://www.oracle.com`

Additional resources

Oracle9*i*AS Portal Information

For all the Oracle9*i*AS Portal documentation visit:

`http://portalcenter.oracle.com`

Oracle Support Services

Technical Support contact information worldwide is listed at:

`http://www.oracle.com/support`

Oracle Store

For U.S.A customers, the Oracle Store is at:

`http://store.oracle.com`

Links to Stores in other countries are provided from this site.

Product Documentation

Product documentation can be found at:

`http://docs.oracle.com`

Education and Training

Training information and worldwide schedules are available from:

`http://education.oracle.com`

Oracle Technology Network

Register with the Oracle Technology Network (OTN) at:

`http://otn.oracle.com`

OTN delivers technical papers, code samples, product documentation, and self-service developer support.

1

Verifying Requirements

Oracle9*i*AS Portal cannot be installed stand-alone. It is installed as part of Oracle9*i* Application Server. You must consult the *Oracle9i Application Server Administrator's Guide* for your particular operating system. As part of a typical Oracle9*i* Application Server installation, the following components are installed:

- Oracle9*i*AS Portal database objects: images, demos, page groups, common services, utilities, required support files, documentation, and Single Sign-On (SSO).
- Oracle9*i* Application Server Web Server: Servlets, mod_plsql, and the Oracle HTTP Server.

See also:

- Oracle HTTP Server documentation in the Oracle9*i* Application Server documentation library for detailed information about Oracle HTTP modules (mods) and the overall Oracle9*i* Application Server architecture
- *Oracle9i Application Server Administrator's Guide* for your particular operating system

1.1 System Requirements

In order to use Oracle9*i*AS Portal, you must have an Oracle9*i* database and Oracle9*i* Application Server running, including the Oracle HTTP Server.

Oracle9*i*AS Portal is packaged and installed with the Oracle9*i* Application Server, in the same Oracle Home location, but in a separate Oracle Home location from the Oracle9*i* database database.

1.1.1 Oracle Databases

Oracle9*i*AS Portal can be installed in the following database:

- Oracle9*i* Release 9.0.1.3 Enterprise and Standard Edition.

  See also: "Administration" section of the *Oracle9iAS Portal Release Notes* for information on the required database patches.

Also, verify that the following conditions exist on your database before configuring Oracle9*i*AS Portal:

- Oracle JServer Option, which includes the Enterprise JavaBeans and CORBA Tools, Java Virtual Machine, and Oracle Java Tools, are installed.
- You have the SYS user password on your database.
- Your database is up and running.

1.1.1.1 Database System resource requirements

The SYSTEM Tablespace should have at least 150 MB of available free space during the Oracle9*i*AS Portal and the Oracle9*i*AS Single Sign-On Server installation. This would ensure faster installation as the extents would not have to be added during installation.

The PORTAL tablespace in the Infrastructure database should have initial storage of at least 75 MB. For the customer database installation, users should make sure that the default tablespace has at least 75 MB of available free space and they should have *auto extend* enabled.

The index tablespace PORTAL_IDX in the Infrastructure database should have initial storage of at least 20 MB. For the customer database installation, users should make sure that the Index tablespace has at least 20 MB of available free space and they should have *auto extend* enabled.

The Document tablespace PORTAL_DOC and Logging tablespace PORTAL_LOG in the Infrastructure database should have initial storage of at least 4 MB. For the customer database installation, users should make sure that the Document and Logging tablespace to have at least 4 MB of available free space and they should have *auto extend* enabled.

The Temporary tablespace PORTAL_TMP in the Infrastructure database should have initial storage of at least 20 MB. For the customer database installation, users should make sure that the Temporary tablespace has at least 20 MB of available free space and they should have *auto extend* enabled.

1.1.2 Oracle Home

You *must* install Oracle9*i* Application Server and its component, Oracle9*i*AS Portal, in a *separate* Oracle home directory from your Oracle9*i* database.

1.1.3 Web Browsers

- Netscape 4.0.8 and 4.72, or higher.
- Microsoft Internet Explorer 4.0.1 with Service Pack 1 and 5.0, or higher.

> **Note:** You may encounter JavaScript errors if you use a browser older than the recommended minimum.

1.1.3.1 Configuring Browser Settings

In order to work optimally, the browser's cache and image settings need to be configured.

1.1.3.2 Cache Settings

Please make sure that your browser cache settings are as follows:

Internet Explorer:

Under Tools->Internet Options -> General Tab -> Settings

```
Check for newer versions of stored pages:
[X] Every visit to the page
[ ] Every time you start Internet Explorer
[ ] Automatically
[ ] Never
```

Netscape:

Under Edit->Preferences->Advanced->Cache

```
Document in cache is compared to document on network:
[ ] Once per session
[X] Every Time
[ ] Never
```

1.1.3.3 Image Settings

Please make sure that pictures/images are automatically loaded as follows:

Internet Explorer:

Under Tools->Internet Options -> Advanced Tab

```
[X] Show Pictures
```

Netscape:

Under Edit->Preferences->Advanced->

```
[X] Automatically Load Images
```

1.1.4 Tablespace Requirements

Since Oracle9*i*AS Portal is installed with Oracle9*i* Application Server, the tablespace amounts are calculated as a whole for all Oracle9*i* Application Server. Refer to the *Oracle9i Application Server Administration Guide* for your particular platform for the documented tablespace requirements.

Note:

- If you are using a FAT file system in your Windows NT/2000 environment, double the amount for the default tablespace.
- During installation, you are prompted to select a tablespace from the database in which to install the Oracle9*i*AS Portal schema and database objects. The default tablespace set by the Oracle9*i*AS Portal Configuration Assistant is USERS.
- Do not allocate additional tablespace for each page group that you create since page groups are part of the Oracle9*i*AS Portal schema.

1.1.5 init.ora Settings

Refer to the *Oracle9i Application Server Administrator's Guide* for your particular operating system for details.

1.1.6 Terminal Settings

On UNIX, set your terminal (using the `set TERM` command) to one of the following types:

```
3151 386 386s 386u 386x
```

```
ansi avx3
dec dgd2 dgd4
hft hftc hp iris ncd
sun sun5 tandm
vt100 vt220
wy150 wy50
xsun xsun5
```

1.1.7 Oracle Text Requirements

To enable Oracle Text searching in Oracle9*i*AS Portal, make sure that the following requirements are met:

- Install and configure the Oracle9*i* database (minimum 9.0.1.3) with the Oracle Text option by running the Oracle Universal Installer (OUI).

1.1.8 Unix Shell Settings

For Unix, you should use csh or ksh for installing Oracle9*i*AS Portal. The Oracle9*i*AS Portal Configuration Assistant (OPCA) uses various Oracle utilities which may not work properly in other shells. The OPCA invokes sub-shells to run various utilities, including SQL*Plus, SQL*Loader, import, and loadjava. You should disable the sourcing to other Oracle homes in the shells before installing Oracle9*i*AS Portal. For example, if your shell sources to an old Oracle home by default, OPCA will create a sub-shell that sources to a different Oracle home, causing the utilities to fail.

2

Post-installation Tasks

This chapter provides information about Oracle9*i*AS Portal after it is installed as part of the Oracle9*i* Application Server and the tasks that you can perform after installation is complete.

> **See also:** *Oracle9i Application Server Administrator's Guide* for information about installing Oracle9*i*AS Portal with Oracle9*i* Application Server.

Specific topics covered include:

- Oracle9iAS Portal Default Schemas
- Oracle9iAS Portal Default Accounts
- Oracle9iAS Portal Default Groups
- Accessing Oracle9iAS Portal in Your Browser
- Configuration Steps for Backward Compatibility
- Configuring Self-registration
- Configuring Directory Synchronization
- SSL Configuration
- Enabling Secure Socket Layer (SSL)
- Configuring Oracle9iAS Portal Security
- Configuring the Oracle Reports Security Portlet
- Configuring WebDAV support for Oracle9iAS Portal Access

2.1 Oracle9*i*AS Portal Default Schemas

If Oracle9*i*AS Portal is installed in the default mode, four Oracle9*i*AS Portal specific schemas are created. The default base schema name is *portal*. This name can be changed at installation time.

Table 2–1 Oracle9iAS Portal default schemas

Schema	Description
portal	The product schema for Oracle9*i*AS Portal and contains the installed Oracle9*i*AS Portal database objects.
*portal*_public	The schema that the Oracle9*i*AS Portal users map to when executing procedures in the Oracle9*i*AS Portal product schema. The schema name is constructed from the base schema with "_ public" appended to it.
*portal*_demo	The schema which is installed with the Oracle9*i*AS Portal demonstration code. The name of this schema is the base schema name with "_demo" appended to it.
*portal*_app	The applications schema for Oracle9*i*AS Portal, which contains the portal applications.

See also: *Oracle9iAS Single Sign-On Administrator's Guide* for descriptions of the default Oracle9*i*AS Single Sign-On Server schemas.

2.2 Oracle9*i*AS Portal Default Accounts

With each Oracle9*i*AS Portal installation, a default set of login accounts is created. If the product is installed in a schema named *portal*, the following default accounts are created:

Table 2–2 Oracle9iAS Portal default accounts

Account	Description
portal	This account is created for the Database Administrator (DBA) with the highest privileges in Oracle9*i*AS Portal.
orcladmin	Similar to portal, this account is granted the highest privileges in Oracle9*i*AS Portal.

Table 2–2 Oracle9iAS Portal default accounts

Account	Description
*portal*_admin	This is the account created for the portal administrator. This account is similar to the DBA account, however, it does not have privileges that provide access to database administration features, such as creating and managing schemas and other database objects.
public	This account is created for public users for unauthenticated sessions. This is the account that all sessions are associated with prior to authentication.

Note: For security reasons, change all the passwords for these accounts after initial login. By default, the password is set to the user name.

See also: Oracle9*i*AS Portal Online Help topics under *Managing users and groups*.

2.3 Oracle9*i*AS Portal Default Groups

The following groups are created at installation time:

Table 2–3 Default Oracle9iAS Portal groups created

Group	Description
DBA	This group has the maximum privilege levels in the system. All global privileges are granted to this group. When this group is installed, it has only one member, the user with the name of the product schema, for example, portal.
PORTAL_ ADMINISTRATORS	This group has most of the global privileges, except for the database-related privileges: ANY_SCHEMA / MANAGE and ANY_SHARED_COMPONENT / MANAGE. This group is comprised of the admin user, portal_admin, and includes the dba group.
PORTAL_DEVELOPERS	This group has privileges to build and manage Oracle9*i*AS Portal components and applications.

Table 2–3 Default Oracle9iAS Portal groups created

Group	Description
PORTLET_PUBLISHERS	This group has the privilege of publishing portlets. Members of this group can create components in the system such as folders, charts, calendars, and so on. This group is initially composed of the portal_administrators group who can then decide which users or groups should be added to this group.
AUTHENTICATED_ USERS	All users that log on to Oracle9*i*AS Portal are added to this group. This is a convenient mechanism to allow logged on users to perform privileged actions. Specified privileges are granted to this group and group membership cannot be changed.
RW_ADMINISTRATOR	This group can CREATE, UPDATE, and DELETE registered report definition files, servers, and printer objects.
RW_BASIC_USER	This group can only run a report if they have been given the privilege to run it.
RW_DEVELOPER	In addition to the privileges of the RW_POWER_USER and RW_BASIC_USER groups, this group can run commands which show the system environment. This group might be assigned to a developer who needs to perform testing and to retrieve detailed error messages.
RW_POWER_USER	In addition to the privileges of the RW_BASIC_USER group, this group can see more detailed error messages.

See also: Oracle9*i*AS Portal Online Help topics under *Managing users and groups*.

2.4 Accessing Oracle9*i*AS Portal in Your Browser

After Oracle9*i*AS Portal is installed, access it by entering the following URL in your browser:

```
http://<hostname>:<portnumber>/pls/<dad>
```

The following table explains the components that make up the URL used to access Oracle9*i*AS Portal.

Table 2–4 URL to enter in browser to access Oracle9iAS Portal

Parameter	Description
hostname	Defines the machine on which you installed Oracle9*i*AS Portal. ■ Enter both the hostname and the fully-qualified domain name. For example, enter *host.domain*.com. ■ This name must also match the `ServerName` parameter in the configuration file, `httpd.conf`, located in: `ORACLE_HOME/Apache/Apache/conf`
portnumber	Defines the port number you specified earlier to access Oracle9*i*AS Portal.
pls	Defines the virtual path and indicates that the request is for a PL/SQL procedure which alerts the Oracle HTTP Server to reroute the request to mod_plsql.
dad	Defines the Database Access Descriptor (DAD) you specified earlier for your Oracle9*i*AS Portal installation. The DAD contains information on how to connect to the database. By default the DAD is 'portal'

See also:

- Section 2.2, "Oracle9iAS Portal Default Accounts"
- *Oracle9i Application Server mod_plsql User's Guide* included in the Oracle9*i* Application Server documentation library.

2.4.1 Simplifying the Full URL of an Oracle9*i*AS Portal Instance

You can simplify the full URL created by the Oracle9*i*AS Portal installation to a more memorable or meaningful URL using the Redirect directive. In this way, end users can access Oracle9*i*AS Portal by entering a simple URL.

By default, the URL for a new Oracle9*i*AS Portal installation requires you to enter:

`http://hostname:portnumber/pls/dad`

You can simplify this URL to:

`http://hostname/redirectpath`

1. Open the Oracle HTTP Server configuration file, `httpd.conf`. This file is located in the following directory:

ORACLE_HOME/Apache/Apache/conf/

2. Enter the redirect path as follows:

   ```
   Redirect /DADnamepath http://hostname:portnumber/pls/dad
   ```

   For example:

   ```
   Redirect /portalhome http://mysite.oracle.com/pls/portal
   ```

   In this example, end users can enter the following:

   ```
   http://mysite.oracle.com/portalhome
   ```

   to access the full URL which is as follows:

   ```
   http://mysite.oracle.com:80/pls/portal
   ```

This technique also works with any valid path that is appended to the URL. For example, if you want to display the Oracle9*i*AS Portal Online Help page group, enter:

```
http://mysite.us.oracle.com/portalhome/url/folder/ONLINE_HELP
```

> **See also:** Oracle9*i*AS Portal Online Help topic: *What are direct access URLs*.

2.5 Configuration Steps for Backward Compatibility

In order for Oracle9*i*AS Portal to be backward compatible with older portals, the following changes need to be made:

1. Add a new DAD similar to what was added in Portal version 3.0.x.
2. Set the DAD configuration parameter PlsqlCompatibilityMode to 1
3. Add the following line to `$IAS_HOME/Apache/Apache/conf/mod_oc4j.conf` (add it next to the other OC4JMount directives):

   ```
   OC4JMount /servlet/*
   ```

4. Setup an application to load the portal.ear file

2.6 Configuring Self-registration

To enable users to create their own portal user accounts, you must configure the self-registration feature.

> **See also:** Oracle9*i*AS Portal Online Help topic *Using Self Registration* for information on how to set up Self Registration.

2.7 Configuring Directory Synchronization

In earlier versions of Oracle Portal, you could authenticate users against an external repository, With the installation of Oracle9*i*AS Portal in Oracle9*i* Application Server Oracle Internet Directory (OID) and the Directory Integration Platform (DIP) is used for all user authentication. OID itself can authenticate against an external repository if that is required.

> **See also:** For an overview of Oracle Internet Directory (OID), refer to the *Oracle Internet Directory Administrator's Guide* in the Oracle9*i* Application Server documentation library.

2.7.1 Setting up a Subscription Profile using oidprovtool

User and Group information is now stored in OID, while Oracle9*i*AS Portal objects and privilege information are still stored in Oracle9*i*AS Portal.

Oracle9*i*AS Portal needs to subscribe to OID, in order to be aware of any changes in OID data. There are two steps involved in setting up a subscription.

OID

On the OID side, a subscription profile needs to be created. A tool named *oidprovtool* is provided for this purpose. It will be located in:

```
ORACLE_HOME/bin
```

The general syntax for invoking this tool is:

```
oidprovtool param1=<param1_value> param2=<param2_value> param3=<param3_value>
```

> **See also:** For a complete example of running the *oidprovtool* and a list of all the Oracle9*i*AS Portal specific parameters, refer to Section B.3, "Using oidprovtool to Create a Subscription Profile".

Once the *oidprovtool* has been run, a subscription profile is created in OID.

Database

On the database where Oracle9*i*AS Portal is installed, log on to SQL*Plus with the appropriate user name and password and enter the following command to set up the OID Preferences in the Oracle9*i*AS Portal schema:

```
EXEC WWSEC_OID.SET_PREFERENCE_VALUE('DIP_INSTALLED', 'Y')
```

This updates the OID preferences and after this you can use the Oracle9*i*AS Portal User Interface in the Global Settings Tab under SSO / OID to update the following two settings:

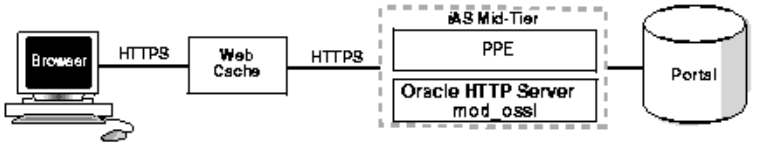
- Time interval. This is the interval at which OID sends notifications of updates to Oracle9*i*AS Portal. Note that it will take the length of the interval to get an update of any changes made in OID. You can adjust the interval to suit your needs.
- Disable / Enable. Here you can disable and enable your subscription profile.

If you need to customize beyond the above mentioned two settings, you can use *oidprovtool*. Refer to the Oracle Internet Directory Administrator's guide in the Oracle9*i* Application Server documentation library, for a complete list of all the possible options for oidprovtool.

2.8 SSL Configuration

The Secure Socket Layer (SSL) enables the securing of HTTP communication between a client and a server. This security is established using a combination of encryption with communication being established through the use of Certificates. Oracle9*i*AS Portal contains several different parts. Web Cache and the Parallel Page Engine, which act as both clients and servers, while the Oracle9*i* Application Server acts simply as a server. Each of these different parts must be configured for SSL usage.

Figure 2–1 Simple Oracle9iAS Portal SSL configuration diagram



In Figure 2-1 each connection over the network is secured using SSL. There are three communication routes involved in any Oracle9*i*AS Portall page rendering. **Note:** For setting up the first two communication routes below, you will need to create an Oracle Wallet.

> **See also:** Oracle Wallet Manager appendix in the *Oracle Internet Directory Administrator's Guide* in the Oracle9*i* Application Server documentation library for more information on managing wallets and certificates.

Browser to Oracle9*i*AS Web Cache

This is the connection that is seen by the end user when requesting a page from Oracle9*i*AS Portal. This connection should be secured using an SSL certificate on the Oracle9*i*AS Web Cache listener.

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library for more information on Oracle9*i*AS Web Cache SSL configuration

Oracle9*i*AS Web Cache to Oracle9*i* Application Server Middle-Tier

Because a browser can be directed to skip past Oracle9*i*AS Web Cache server, if the proper port numbers are provided, it is important to configure the Oracle9*i* Application Server for SSL communication.

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library for more information on Server SSL Configuration

Parallel Page Engine (PPE) to Oracle9*i*AS Web Cache

This communication path is already secure based upon the securing of Oracle9*i*AS Web Cache in the first option above, however there are a couple of configuration items which must be changed for the PPE to recognize the use of SSL.

> **See also:** Section 2.8.1, "Securing Ports to Use Certificates and HTTPS"

2.8.1 Securing Ports to Use Certificates and HTTPS

With HTTPS, you use certificates for ports to increase security.

Note: The Certificate Authority (CA) file is the base signature file for the certificate file you have purchased. This file validates the certificate you are using. It informs clients that they can trust the certificate they have received. You require a CA file for any type of certificate you use.

To set this up, edit the OC4J web.xml file. The web.xml file can be found in the following location:

```
ORACLE_HOME/j2ee/OC4J/applications/portal/portal/WEB_INF/web.xml
```

You must set up HTTPS such that it is used by all ports at all times. The Parallel Page Engine must be aware of which port(s) are operating under HTTPS.

To do this, add the following XML block to the web.xml file:

```
<init-param>
<param-name>httpsports</param-name>
<param-value>433:444</param-value>
</init-param>
```

Where the port numbers 433 and 444 are replaced by your HTTPS port configuration. Your server only needs to have one port, but two are shown in the above example to show the syntax used for multiple entries. Each port in this list operates using the HTTPS protocol, and must have a certificate created on the Oracle HTTP Server for that port.

See also:

- Section 6.2, "Parallel Page Engine Configuration"
- Oracle9*i*AS Portal Online Help topic: *Configuring the Oracle9i*AS Single Sign-On Server *for LDAP user authentication.*

2.9 Enabling Secure Socket Layer (SSL)

Oracle9*i*AS Portal and the Oracle9*i*AS Single Sign-On Server can be configured to run in HTTPS mode if your portal requires increased security. For optimal performance, you can also choose to have a mixed configuration where Oracle9*i*AS Portal is run in HTTP mode and the Oracle9*i*AS Single Sign-On Server is run in HTTPS mode.

Secure Socket Layer (SSL) is responsible for securing Web HTTP communication between a browser and a Web server in plain HTTP over SSL (named HTTPS). Enabling SSL to work with the Oracle HTTP Server is handled by the `mod_ssl` package which is provided with the Oracle HTTP Server. It uses the URL scheme HTTPS rather than HTTP and a different server port.

Note: You must be the portal administrator to enable or disable Secure Socket Layer (SSL) in Oracle9*i*AS Portal and on the Oracle9*i*AS Single Sign-On Server.

See also: *Oracle9i Application Server Administrator's Guide* for more information on enabling SSL on the server.

2.9.1 Setting Oracle9*i*AS Single Sign-On Server Query Path URL

Oracle9*i*AS Portal maintains the URL prefix of the Oracle9*i*AS Single Sign-On Server which accesses certain information through HTTP calls from the database, using the `UTL_HTTP` package. These calls must be done through HTTP rather than HTTPS.

Thus, if Oracle9*i*AS Portal and the Oracle9*i*AS Single Sign-On Server are configured to use HTTPS, access to an HTTP port on the Oracle9*i*AS Single Sign-On Server is still required to support these interfaces. The calls made across this interface are required for the following reasons:

- Obtain the list of external applications to allow the external applications portlet to customize.
- Perform the mapping of the Single Sign-On user name to the external application user name.
- Present an interface to verify the existence of a user.

To set this URL prefix, which is called the Oracle9*i*AS Single Sign-On Server Query Path URL, complete these steps:

1. Log on to Oracle9*i*AS Portal as the portal administrator.
2. Click the **Administer** tab.
3. Click **Global Settings** in the Services Portlet.

4. Scroll down to the section on Oracle9*i*AS Single Sign-On Server, and edit the *Query Path URL*. Set this field to an HTTP URL for the Oracle9*i*AS Single Sign-On Server.

2.9.2 Adding SSO Enabler Configuration Entries for HTTPS Mode

If you are using SSL, the default port is 443. With Oracle Portal versions prior to 3.0.8, you need to create two enabler configuration entries, and two corresponding partner configuration entries on the Oracle9*i*AS Single Sign-On Server. Specify the :443 port for one entry, and exclude it for the additional entry.

To add the additional entry, follow the basic procedure of adding the partner entry on the Oracle9*i*AS Single Sign-On Server Server using the Oracle9*i*AS Single Sign-On Server Server Administration user interface, and then add the configuration entry on the Oracle Portal side by running the Oracle9*i*AS Portal Configuration Assistant in the SSOPARTNERCONFIG mode.

Note: This step is only required to support Netscape browsers. Microsoft Internet Explorer does not require this step.

2.9.3 Configuring HTTPS with Virtual Hosts

If you want to setup a virtual host, it can be done in one of two ways:

- Through an IP address (for example, `123.1.3.2`)
- Through an IP name (for example, `server.oracle.com`)

When the IP name is used, several aliases use the same IP address. In this case, Oracle HTTP Server (or any browser supporting virtual name addresses) looks at the Host field in the HTTP request and determines which of the virtual addresses should be emulated.

However, when SSL is used, the IP name is encrypted. This causes the problem, because the software does not know which decryption key to use since the keys differ by virtual name. If there were 1000 separate virtual addresses supported, then on average the software would try 500 different keys to determine which key to use to decode the message. This is not practical, at least for performance reasons.

Note:

- It is more difficult to configure virtual hosts to use HTTPS since the SSL encryption prevents virtual hosts from being resolved in the way that it is done in non-SSL mode.
- There are some workarounds from which to choose. One is to only use virtual names on the home page and other pages where you do not need protection.

2.9.3.1 SSL Protection Pages

- Obtain an IP address for each virtual host and then obtain a TCP/IP card that can handle multiple IP addresses, one for each virtual host.
- Use one IP address with different port numbers for each virtual host.
- Use one IP address but use different directories, one for each "virtual host" (for example, `https://ssladdress.com/virtualname1/<page desired>`).

2.10 Configuring Oracle9*i*AS Portal Security

See also:

- *Oracle9i* Application Server *Security Guide* in the Oracle9*i* Application Server documentation library.
- Section 3.2.7, "SYSOBJECTS OPCA Mode"
- Section B.5, "Using the secupoid.sql script"
- Section B.6, "Using the secjsdom.sql script"

2.11 Configuring the Oracle Reports Security Portlet

The Oracle Reports Security Portlet is installed as part of the Oracle9*i*AS Portal installation. Depending on the edition of the Oracle9*i* Application Server installed, the Reports Security Portlet may be hidden. To show the Reports Security Portlet:

1. Log on to Oracle9*i*AS Portal as the portal administrator.
2. Click the **Administer** tab.
3. Click **Edit**.

4. Find the **Oracle Reports Security Portlet.**
5. Click on **Actions**.
6. Select the **Show** action and click OK.

2.12 Configuring WebDAV support for Oracle9*i*AS Portal Access

WebDAV is a protocol extension to HTTP 1.1 that supports distributed authoring and versioning. With WebDAV, the Internet becomes a transparent read and write medium, where content can be checked out, edited, and checked in to a URL address. mod_dav is an implementation of the WebDAV specification. The standard mod_dav implementation supports read and write access to files.

The term OraDAV refers to the capabilities available through the mod_oradav module. mod_oradav is the Oracle module that is an extended implementation of mod_dav, and is integrated with the Oracle HTTP Server. mod_oradav can read and write to local files, but also to an Oracle database. The Oracle database must have an OraDAV driver installed. mod_oradav calls this driver to map WebDAV activity to database activity. mod_oradav enables WebDAV clients to connect to an Oracle database, read and write content, and query and lock documents in various schemas.

See also:

Oracle HTTP Server Administration Guide for information about Web DAV, oraDAV and how to set up mod_oradav.

Oracle9*i*AS Portal online help topic "*Accessing Oracle9iAS Portal from WebDAV clients*".

When Oracle9*i* Application Server is installed, all required OraDAV parameters are set with values that are designed to enable Oracle database content to be accessed through a web browser or WebDAV client. If necessary, you can later modify the values for required parameters and specify values for optional parameters, if the default values do not meet your needs.

The OraDAV parameters are stored in the *oradav.conf* file and start with DAV and DAVParam. These parameters are specified within a <*Location*> directive. The *oradav.conf* file is included in the *httpd.conf* file in an include statement.

After Oracle9*i*AS Portal has been installed as part of the Oracle9*i* Application Server installation, the *oradav.conf* file should be populated with a <Location> directive which points to the portal schema. In the following example, the location */dav_*

portal/portal will be OraDAV-enabled and will (once populated with the correct values) connect to the portal schema so that users can use WebDAV clients to access portal data.

Example 2–1 Configuration Parameters for Portal Access

```
<Location /dav_portal/portal>
   DAV Oracle
   DAVParam ORACONNECT dbhost:dbport:dbsid
   DAVParam ORAUSER portal_schema
   DAVParam ORAPASSWORD portal_schema_password
   DAVParam ORAPACKAGENAME portal_schema.wwdav_api_driver
</Location>
```

By default, the Oracle9*i*AS Portal DAV URL is:

`http://`*`hostname`*`:`*`port`*`/dav_portal/`*`dadname/`*

In the above example, the DAD name is *portal* (the default). The directive created for the Location is *dav_portal/portal* and the DAV URL will be:

```
http://dbhost:dbport/dav_portal/portal/
```

Note: When you add a new DAD, using the Oracle9*i*AS Portal Configuration Assistant (OPCA), without specifying the username and password, or if you change the Portal database schema username or password, using SQL*Plus, you will need to update the *httpd.conf* and *oradav.conf* files manually.

3

Configuring Oracle9*i*AS Portal using OPCA

This chapter covers:

- The Oracle9iAS Portal Configuration Assistant (OPCA)
- OPCA Modes

3.1 The Oracle9*i*AS Portal Configuration Assistant (OPCA)

The Oracle9*i*AS Portal Configuration Assistant (OPCA) is a java based configuration tool for installing and configuring the Oracle9*i*AS Portal repository. In a typical Oracle9*i* Application Server install the OPCA is invoked by the Oracle Universal Installer (OUI) in the post installation phase. The OPCA can also be invoked in the stand-alone mode.

3.1.1 OPCA and the Oracle9*i* Application Server Installation

In an Oracle9*i* Application Server installation, using OUI, the Oracle9*i*AS Portal install is performed in two phases. The Oracle9*i* Application Server *Infrastructure* install type loads the Portal repository in the database. The Oracle9*i* Application Server middle-tier install type performs the required configuration of the middle-tier, linking it with the Portal repository.

The OPCA is installed with the Oracle9*i* Application Server and is located in the directory

```
<iAS_Home>/assistants/opca
```

The scripts *launch.sh* and *launch.bat* are located at:

```
<iAS_Home>/assistants/opca
```

They are used by the Oracle Universal Installer (OUI) to invoke the OPCA on the UNIX and Windows NT/2000 platform respectively.

Note: The launch scripts should not be used by users to invoke the OPCA. The only mode that is currently supported in the launch scripts is the MIDTIER mode with -typical installation type, most of the parameters of this mode would be obtained by the Repository Access API's within OPCA.

3.1.2 Using OPCA in Stand-alone mode

Starting with the Oracle9*i* Application Server release, Oracle9*i*AS Portal provides a command line script to invoke the OPCA in stand-alone mode. The scripts ptlasst.csh (UNIX) and ptlasst.bat (Windows NT/2000) are located at `<iAS_Home>/assistants` dir and can be used to run the OPCA stand-alone in different modes. To start the Oracle9*i*AS Portal Configuration Assistant, perform the following steps:

On Windows NT/2000:

First, set the ORACLE_HOME to the Oracle9*i* Application Server Home and JRE_HOME to the JRE home. You can then start the OPCA from the command line by navigating to the <iAS_Home>/assistants directory and using the command:

```
ptlasst.bat [parameters]
```

On UNIX:

First set the ORACLE_HOME to the Oracle9*i* Application Server Home.You can start the OPCA from the command line by navigating to the `iAS_Home/assistants` directory and using the command:

```
ptlasst.csh [parameters]
```

The ptlasst scripts launches the java based configuration program *opca* and can be launched without passing in any parameters.

3.2 OPCA Modes

For a description of all the OPCA modes, check the table below:

Mode	Description
PORTAL OPCA Mode	This mode installs the Portal repository in the target database. No information from the midtier is required. This mode can be used to create a pre-seeded Portal in the infrastructure database
MIDTIER OPCA Mode	This mode populates and configures the midtier for the Portal repository. Associating the SSO Server, Portal Repository, Oracle9*i*AS Web Cache and the Portal Repository as well as associating the *i*ASW with the Portal Repository and creating the OID entries for Portal This mode assumes that OID, Oracle9*i*AS Portal and the Oracle9*i*AS Single Sign-On Server is installed in the database. Additionally, Oracle9*i*AS Web Cache and *i*ASW need to be fully configured.
SSO OPCA Mode	This mode installs a stand-alone SSO server in the target database. This mode can be used to create a pre seeded SSO in the infrastructure database.
SSOPARTNERCONFIG OPCA Mode	This mode allows a Portal partner application to add to, modify, or delete, it's configuration information relating to the SSO server. This mode appends to the existing configurations.
LANGUAGE OPCA Mode	This mode installs the strings for a specific language in the Portal repository.
ALL OPCA Mode	This mode comprises of the PORTAL, SSO and MIDTIER mode.
SYSOBJECTS OPCA Mode	Installs Portal and SSO required SYS schema dependencies.

Note: The scripts winstall, winstall.bat, ssodatan, ssodata.bat, langinst.csh, and langinst.bat are deprecated and their functionality is now handled by OPCA through the ptlasst scripts.

3.2.1 PORTAL OPCA Mode

This mode installs the Portal repository in the target database. No information from the middle-tier is required. This mode can be used to create a pre-seeded Portal in the infrastructure database.

Action

Installs Portal Repository in the database with no middle-tier wiring and should be used for the Portal seed database creation.

Usage Example

Full Usage

```
ptlasst.csh -i typical -mode PORTAL -s portal -c myhost.domain.com:1521:mySID -p
change_on_install -u users -t temp -d users -l users -in users -demo  -silent
-verbose  -report -owa
```

Table 3–1 List of supported parameters for the PORTAL mode

Parameter	Description
`-i`	Install type. This can be set to *typical* and *custom*. In the typical mode, the OPCA takes all the user input passed in on the command line and only a progress bar will be visible. In the custom mode, the OPCA user interface is invoked, which includes all the wizards that prompt for user input, containing the values that are passed in on the command line if they are specified. Note that the Repository Access API's are only used only in the typical mode. Note also that the -silent parameter takes precedence over the -i parameter.
`-mode`	install mode. The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-s`	Portal schema name.
`-c`	Connect string to the target database. The format should be *hostname:port:sid*.
`-p`	SYS password for the target database.
`-u`	Default tablespace.
`-t`	Temporary tablespace.
`-d`	Document tablespace.
`-l`	Logging tablespace.
`-in`	Index tablespace.
`-demo`	Installs the portlet builder demo components.
`-silent`	Runs the OPCA in the silent mode.
`-verbose`	This parameter enables logging in detail mode. The OPCA install continues even if there are errors in the log file.

Table 3–1 List of supported parameters for the PORTAL mode

Parameter	Description
`-report`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal.
`-owa`	This parameter installs the PL/SQL Web Toolkit and other SYS schema packages. This need to be installed just once in the database.

3.2.2 MIDTIER OPCA Mode

Action

- Associates the Oracle9*i*AS Single Sign-On Server and the Oracle9*i*AS Portal repository.
- Associates the Oracle9*i*AS Web Cache and the Oracle9*i*AS Portal Repository.
- Associates the *i*ASW and the Oracle9*i*AS Portal Repository.

Assumptions

- Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server are already installed.
- OID, Oracle9*i*AS Single Sign-On Server and Oracle HTTP server are up and running.

Usage example

Full Usage

```
ptlasst.csh -i typical -mode MIDTIER -s portal -sp portal -c
myhost.domain.com:1521:mySID -sdad portal -o orasso -op orasso -odad websso_sso
-host myApache.domain.com -port 7777 -silent -verbose-ldap_h myOID.domain.com
-ldap_p 389 -ldap_d cn=orcladmin -ldap_w welcome -portal_only -sso_only -chost
myHostname.domain.com -cport_i 8001 -cport_a 8000 -sso_c
myhost.domain.com:1521:mySID -sso_h myApache.domain.com -sso_p 7777
-ultrasearch-oh /home/oracle -mc false -mi true
```

Note:

- `-portal_only` and `-sso_only` should not be passed together.
- See Section , "Problem: OPCA install hangs at Oracle Ultra Search phase" for possible issues that might occur during the installation.

Usage example

Basic Usage (Typical mode)

```
ptlasst.csh -i typical -mode MIDTIER  -host myApache.domain.com -port 7777
-silent -verbose  -ldap_d cn=orcladmin -ldap_w welcome -oh /home/oracle -mc
false -mi true
```

Usage example

Basic Usage (Custom mode)

```
ptlasst.csh -i custom -mode MIDTIER -s portal -sp portal -c
myhost.domain.com:1521:mySID -sdad portal -o orasso -op orasso -odad websso_sso
-host myApache.domain.com -port 7777 -silent -verbose -ldap_h myOID.domain.com
-ldap_p 389 -ldap_d cn=orcladmin -ldap_w welcome -chost myHostname.domain.com
-cport_i 8001 -cport_a 8000 -sso_c myhost.domain.com:1521:mySID -sso_h
myApache.domain.com -sso_p 7777 -ultrasearch -oh /home/oracle -mc false -mi true
```

Usage example

Portal only configuration

```
ptlasst.csh -i typical -mode MIDTIER -s portal -sp portal -c
myhost.domain.com:1521:mySID -sdad portal -o orasso -op orasso -odad websso_sso
-host myApache.domain.com -port 7777 -silent -verbose -ldap_h myOID.domain.com
-ldap_p 389 -ldap_d cn=orcladmin -ldap_w welcome -portal_only  -chost
myHostname.domain.com -cport_i 8001 -cport_a 8000 -sso_c
myhost.domain.com:1521:mySID -sso_h myApache.domain.com -sso_p 7777
-ultrasearch-oh /home/oracle -mc false -mi true
```

Usage example

SSO Server only configuration

```
ptlasst.csh -i typical -mode MIDTIER -o orasso -op orasso -odad websso_sso
-silent -verbose -ldap_h myOID.domain.com -ldap_p 389 -ldap_d cn=orcladmin
-ldap_w welcome -sso_only -sso_c myhost.domain.com:1521:mySID -sso_h
myApache.domain.com -sso_p 7777  -oh /home/oracle -mc false -mi true
```

Table 3–2 List of supported parameters for the MIDTIER mode.

Parameter	Description
`-i`	Install type. This can be set to *typical* and *custom*. In the typical mode, the OPCA takes all the user input passed in on the command line and only a progress bar will be visible. In the custom mode, the OPCA user interface is invoked, which includes all the wizards that prompt for user input, containing the values that are passed in on the command line if they are specified. Note that the Repository Access API's are only used only in the typical mode. Note also that the -silent parameter takes precedence over the -i parameter.
`-mode`	**Mandatory** install mode. The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-mi`	**Mandatory** Midtier Installation. This includes the Oracle9*i*AS Portal OID and Oracle9*i*AS Single Sign-On Server integration. Default value: true
`-mc`	**Mandatory** middle-tier Configuration. This includes the Oracle9*i*AS Portal OC4J deployment and Oracle Enterprise Manager configuration. Default value: false
`-s`	Portal schema name. Default value: portal.
`-sp`	Portal schema password. Default value: portal.
`-c`	**Mandatory** connect string to the target Portal database. The format should be hostname:port:sid

Table 3–2 List of supported parameters for the MIDTIER mode.

Parameter	Description
`-sdad`	-sdad Portal Schema DAD name. Default value: portal
`-o`	Oracle9*i*AS Single Sign-On Server schema name. Default value: orasso.
`-op`	Oracle9*i*AS Single Sign-On Server password. Default value: orasso.
`-odad`	Oracle9*i*AS Single Sign-On Server DAD name. Default value: orasso.
`-host`	**Mandatory** HTTP server host name used for Oracle9*i*AS Portal.
`-port`	**Mandatory** HTTP server Port number used for Oracle9*i*AS Portal.
`-chost`	Oracle9*i*AS Web Cache host.
`-cport_i`	Oracle9*i*AS Web Cache invalidation port.
`-cport_a`	Oracle9*i*AS Web Cache administration port.
`-ldap_h`	**Mandatory** Host name of the OID server.
`-ldap_p`	**Mandatory** Port number of the OID server.
`-ldap_d`	**Mandatory** Administration DN.
`-ldap_w`	**Mandatory** Password for DN.
`-sso_c`	Connect string for the SSO database. The format should be *hostname:port:sid*.
`-sso_h`	HTTP server host name used for Oracle9*i*AS Single Sign-On Server.
`-sso_p`	HTTP server port number used for SSP.
`-ultrasearch`	Ultrasearch Integration.
`-oh`	Oracle Home.
`-portal_only`	Middle-tier Configuration for Oracle9*i*AS Portal only.
`-sso_only`	Middle-tier configuration for Oracle9*i*AS Single Sign-On Server only.

Table 3–2 List of supported parameters for the MIDTIER mode.

Parameter	Description
`-silent`	Runs the OPCA in silent mode.
`-verbose`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal. If this parameter is not set, logging information will be brief and the OPCA aborts the installation if it encounters any kind of ORA-,PLS- or SP2 errors.

See also: *Oracle9i Application Server Administrator's Guide* for more information on Oracle Enterprise Manager.

3.2.3 SSO OPCA Mode

Action

Installs a stand-alone Oracle9*i*AS Single Sign-On Server.

Assumptions

- OID has been installed and fully configured.
- The middle-tier is installed.
- The Oracle9*i*AS Single Sign-On Server DAD has been created.

Usage example

SSO Server Install with repository and middle-tier

```
ptlasst.csh -i typical -mode SSO -s orasso -sso_c myhost.domain.com:1521:mySID
-p change_on_install  -o orasso -odad orasso -sso_h myApache.domain.com -sso_p
7777  -u users -t temp -d users -l users -in users -silent -verbose  -owa -ldap_
h myOID.domain.com -ldap_p 389 -ldap_d cn=orcladmin -ldap_w welcome -ssotype all
```

Usage example

SSO Server repository only install

```
ptlasst.csh -i typical -mode SSO -s orasso -sso_c myhost.domain.com:1521:mySID
-p change_on_install -o orasso -u users -t temp -d users -l users -in users
-silent -verbose -owa -ssotype repository
```

Usage example

SSO Server middle-tier only configuration

```
ptlasst.csh -i typical -mode SSO -s orasso -sso_c myhost.domain.com:1521:mySID
-p change_on_install  -o orasso -odad orasso -sso_h myApache.domain.com -sso_p
7777  -u users -t temp -d users -l users -in users -silent -verbose -owa -ldap_h
myOID.domain.com -ldap_p 389 -ldap_d cn=orcladmin -ldap_w welcome -ssotype
midtier
```

Table 3–3 List of supported parameters for the SSO mode

Parameter	Description
`-i`	Install type. This can be set to *typical* and *custom*. In the typical mode, the OPCA takes all the user input passed in on the command line and only a progress bar will be visible. In the custom mode, the OPCA user interface is invoked, which includes all the wizards that prompt for user input, containing the values that are passed in on the command line if they are specified. Note that the Repository Access API's are only used only in the typical mode. Note also that the -silent parameter takes precedence over the -i parameter.
`-mode`	**Mandatory** install mode. The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-sso_c`	**Mandatory** connect string to the target database. The format should be *hostname:port:sid*.
`-sso_h`	HTTP server hostname for the Oracle9*i*AS Single Sign-On Server.
`-sso_p`	HTTP server port number for the Oracle9*i*AS Single Sign-On Server.
`-ssotype`	**Mandatory** option with three possible values: ALL, REPOSITORY, MIDTIER.
`-p`	SYS password for the target database.
`-o`	Oracle9*i*AS Single Sign-On Server server schema name.
`-odad`	Oracle9*i*AS Single Sign-On Server DAD name.
`-u`	Default tablespace.

Table 3–3 List of supported parameters for the SSO mode

Parameter	Description
`-d`	Document tablespace.
`-l`	Logging tablespace.
`-in`	Index tablespace.
`-silent`	Runs the OPCA in silent mode.
`-verbose`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal. If this parameter is not set, logging information will be brief and the OPCA aborts the installation if it encounters any kind of ORA-,PLS- or SP2 errors.
`-owa`	This parameter installs the PL/SQL Web Toolkit and other SYS schema packages. This need to be installed just once in the database.
`-ldap_h`	Host name of the OID server.
`-ldap_p`	Port number of the OID server.
`-ldap_d`	Administration DN.
`-ldap_w`	Password for DN.

3.2.4 SSOPARTNERCONFIG OPCA Mode

Action

Allows an Oracle9*i*AS Portal partner application to add to, modify, or delete, it's configuration information relating to the Oracle9*i*AS Single Sign-On Server. This mode *appends* to the existing configurations.

Assumptions

- Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server server are installed.
- The Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server DADs are created.
- The Oracle HTTP Server is installed, configured and running.

Usage example

Add or modify a Partner application

```
ptlasst.csh -i typical -mode SSOPARTNERCONFIG -s_id mySiteId -s_tkn mySiteToken
-s_key  2145432 -s_cookie v1.2 -ps websso_ps -pp websso_ps -pd myPwdDBLink -s
portal -sp portal -c myhost.domain.com:1521:mySID -sdad portal -odad websso
-host myApache.domain.com -port 7777 -silent -verbose -sso_c
myhost.domain.com:1521:mySID -sso_h myApache.domain.com -sso_p 7777 -p_tns
orasso_ps -s_tns portal
```

Usage example

Delete a Partner application

```
ptlasst.csh -i typical -mode SSOPARTNERCONFIG -s portal -sp portal -c
myhost.domain.com:1521:mySID -silent -verbose -remove portalHost
```

Table 3–4 List of supported parameters for the SSOPARTNERCONFIG mode

Parameter	Description
`-i`	Install type. This can be set to *typical* and *custom*. In the typical mode, the OPCA takes all the user input passed in on the command line and only a progress bar will be visible. In the custom mode, the OPCA user interface is invoked, which includes all the wizards that prompt for user input, containing the values that are passed in on the command line if they are specified. Note that the Repository Access API's are only used only in the typical mode. Note also that the -silent parameter takes precedence over the -i parameter.
`-mode`	**Mandatory** install mode. The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-s_id`	**Mandatory** Site ID corresponding to the Oracle9*i*AS Single Sign-On Server's partner application configuration entry for this Oracle9*i*AS Portal instance.
`-s_tkn`	**Mandatory** Site token corresponding to the Oracle9*i*AS Single Sign-On Server's partner application configuration entry for this Oracle9*i*AS Portal instance.
`-s_key`	**Mandatory** Encryption Key corresponding to the Oracle9*i*AS Single Sign-On Server partner application configuration entry for this Oracle9*i*AS Portal instance.

Table 3–4 List of supported parameters for the SSOPARTNERCONFIG mode

Parameter	Description
`-s_cookie`	**Mandatory** Cookie version being used by the Oracle9*i*AS Single Sign-On Server.
`-ps`	**Mandatory** Password store schema.
`-pp`	**Mandatory** Password store password.
`-pd`	**Mandatory** Database link from Portal schema to password store.
`-s`	**Mandatory** Portal schema name.
`-sp`	Portal schema password.
`-c`	Connect string to the target database. The format should be *`hostname:port:sid`*.
`-sdad`	DAD for the Oracle9*i*AS Portal schema.
`-odad`	DAD for the Oracle9*i*AS Single Sign-On Server schema.
`-host`	HTTP server hostname for Oracle9*i*AS Portal.
`-port`	HTTP server port number for Oracle9*i*AS Portal.
`-silent`	Runs the OPCA in silent mode.
`-verbose`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal. If this parameter is not set, logging information will be brief and the OPCA aborts the installation if it encounters any kind of ORA-,PLS- or SP2 errors.
`-sso_c`	Connect string to the target database. The format should be *hostname:port:sid*.
`-sso_h`	HTTP server hostname for Oracle9*i*AS Single Sign-On Server.
`-sso_p`	HTTP server portnumber for Oracle9*i*AS Single Sign-On Server.
`-p_tns`	**Mandatory** TNS connect string to Password Store.
`-s_tns`	**Mandatory** TNS connect string to Portal.
`-remove`	Removes the enabler configuration entry associated with the specified Portal.

3.2.5 LANGUAGE OPCA Mode

Action

Installs the strings for a specific language in the Oracle9*i*AS Portal repository or Oracle9*i*AS Single Sign-On Server or both.

Oracle9*i*AS Portal is translated into 29 different languages. This allows developers to work in their own language when they build portals. In addition, the self-service content management supports multiple languages so that end users can provide documents and other content in different languages. Those who view the content can see the version that corresponds to the language setting in the browser or to the language they have selected in the set language portlet.

The table below shows the languages that are available for Oracle9*i*AS Portal:

Table 3–5 Oracle9iAS Portal Languages

Language	Language Abbreviation
Arabic	ar
Czech	cs
German	d
Danish	dk
Spanish	e
Greek	el
Latin American Spanish	esa
French	f
Canadian French	frc
Hebrew	iw
Hungarian	hu
Italian	i
Japanese	ja
Korean	ko
Norwegian	n
Dutch	nl
Polish	pl

Table 3–5 Oracle9iAS Portal Languages (Cont.)

Language	Language Abbreviation
Portuguese	pt
Brazilian Portuguese	ptb
Romanian	ro
Russian	ru
Swedish	s
Finnish	sf
Slovak	sk
Turkish	tr
Thai	th
Simplified Chinese	zhs
Traditional Chinese	zht

To install languages when you install Oracle9*i*AS Portal, run OPCA in the LANGUAGE mode. Note that you must run the `ptlasst.csh` script with `-mode LANGUAGE` for each language that you want Oracle9*i*AS Portal to support.

Running the `ptlasst.csh` script with `-mode LANGUAGE` invokes the Oracle9*i*AS Portal Configuration Assistant (OPCA) in the silent mode to install the language. Usage information on the `ptlasst.csh` script is generated by running the script without any parameters.

Assumptions

Portal and SSO repository has been already installed.

Usage example

Full usage

```
ptlasst.csh -mode LANGUAGE -s portal -sp portal -o orasso -op orasso -c
myhost.domain.com:1521:mySID -lang us -available-silent-verbose-sso_c
myhost.domain.com:1521:mySID
```

Usage example

Portal and SSO Server are installed on the same database

```
ptlasst.csh -mode LANGUAGE -s portal -sp portal -o orasso -op orasso -c
myhost.domain.com:1521:mySID -lang us -available-silent-verbose
```

Usage example

Language requirement is for the Portal Repository only

```
ptlasst.csh -mode LANGUAGE -s portal -sp portal -c myhost.domain.com:1521:mySID
-lang us -available -silent -m portal -verbose
```

Usage example

Language requirement is for the SSO Repository only.

```
ptlasst.csh-mode LANGUAGE -o orasso -op orasso -sso_c
myhost.domain.com:1521:mySID -lang us -available -silent -m sso -verbose
```

Table 3–6 List of supported Parameters for the LANGUAGE mode

Parameter	Definition
`-mode`	**Mandatory** install mode. The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-s`	Portal schema name. Default value: `portal`.
`-sp`	Portal schema password. Default value: `portal`.
`-c`	**Mandatory** connect string to the target database. The format should be *`hostname:port:sid`*.
`-o`	Oracle9*i*AS Single Sign-On Server schema name. Default value: `orasso`.
`-op`	Oracle9*i*AS Single Sign-On Server password. Default value: `orasso`.

Table 3–6 List of supported Parameters for the LANGUAGE mode (Cont.)

Parameter	Definition
`-sso_c`	Connect string to the target database where Oracle9*i*AS Single Sign-On Server is installed. The format should be `hostname:port:sid`.
	Note: If the Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server use the same database, argument `-c` would take care of both the Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server connections.
`-lang`	Abbreviation for the language to install.
	Default value: `us`
`-m`	If `sso`, translations are only for Oracle9*i*AS Single Sign-On Server repository. If `portal`, translations are only for Oracle9*i*AS Portal repository. If not specified, translations are installed for both.
`-available`	Sets whether the language will be available for user translation.
`-silent`	Runs the OPCA in the silent mode.
	Default value: `TRUE`.
`-verbose`	Enables the logging in detail mode. Even if there are errors in the log file, the OPCA install would continue.
	If this parameter is not set, logging information will be brief and the OPCA aborts the installation if it encounters any kind of ORA-,PLS- or SP2 errors.
	Default value: `TRUE`.

Note:

- When you configure bidirectional languages (Arabic and Hebrew), you must also execute the `ORACLE_HOME/ora9ias/portal30/admin/plsql/nlsres/imginst.sql` script as the Portal schema owner.
- The character set for `mod_plsql` must be the same as the customer database character set.

See also: *Oracle9i Application Server Globalization Support Guide* in the Oracle9i Application Server documentation library.

3.2.6 ALL OPCA Mode

Action

- Installs Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server repository.
- Configures the Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server middle-tier.

Assumptions

- OID has been installed and fully configured.
- Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server are to be installed in the same database.
- Middle-tier is already installed, configured and running.

Usage example

Full usage

```
ptlasst.csh -i typical -mode ALL -s portal -c myhost.domain.com:1521:mySID -p
change_on_install -sdad portal -o orasso -odad websso_sso -host
myApache.domain.com -port 7777 -chost myHostname.domain.com -cport_i 8001
-cport_a 8000 -u users -t temp -d users -l users -in users -demo  -silent
-verbose -report  -owa -ldap_h myOID.domain.com -ldap_p 389 -ldap_d cn=orcladmin
-ldap_w welcome -ultrasearch -sso_h myApache.domain.com -sso_p 7777 -oh
/home/oracle -mc false -mi true
```

Usage example

Portal and SSO Server on the same database using the same HTTP server

```
ptlasst.csh -i typical -mode ALL -s portal -c myhost.domain.com:1521:mySID -p
change_on_install -sdad portal -o orasso -odad websso_sso -host
myApache.domain.com -port 7777 -chost myHostname.domain.com -cport_i 8001
-cport_a 8000 -u users -t temp -d users -l users -in users -demo  -silent
-verbose -report  -owa -ldap_h myOID.domain.com -ldap_p 389 -ldap_d cn=orcladmin
-ldap_w welcome -ultrasearch-oh /home/oracle -mc false -mi true
```

Table 3–7 List of supported parameters for the ALL mode

Parameter	Description
`-i`	Install type. This can be set to *typical* and *custom*. In the typical mode, the OPCA takes all the user input passed in on the command line and only a progress bar will be visible. In the custom mode, the OPCA user interface is invoked, which includes all the wizards that prompt for user input, containing the values that are passed in on the command line if they are specified. Note that the Repository Access API's are only used only in the typical mode. Note also that the -silent parameter takes precedence over the -i parameter.
`-mode`	**Mandatory** install mode The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-s`	Portal schema name.
`-c`	**Mandatory** connect string to the target database. The format should be *hostname:port:sid*.
`-p`	**Mandatory** SYS password of the target database.
`-sdad`	DAD for the Portal repository.
`-o`	Oracle9*i*AS Single Sign-On Server schema name.
`-odad`	DAD for the Oracle9*i*AS Single Sign-On Server server.
`-host`	**Mandatory** HTTP server Hostname for Oracle9*i*AS Portal.
`-port`	**Mandatory** HTTP server Port for Oracle9*i*AS Portal.
`-u`	Default tablespace.
`-t`	Temporary tablespace.
`-d`	Document tablespace.
`-l`	Logging tablespace.
`-in`	Index tablespace.
`-chost`	Oracle9*i*AS Web Cache host.
`-cport_i`	Oracle9*i*AS Web Cache invalidation port.
`-cport_a`	Oracle9*i*AS Web Cache administration port.

Table 3–7 List of supported parameters for the ALL mode

Parameter	Description
`-demo`	Portlet builder demo components will be installed.
`-silent`	Runs the OPCA in silent mode.
`-verbose`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal.
	If this parameter is not set, logging information will be brief and the OPCA aborts the installation if it encounters any kind of ORA-,PLS- or SP2 errors.
`-report`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal.
`-owa`	This parameter installs the PL/SQL Web Toolkit and other SYS schema packages. This need to be installed just once in the database.
`-ldap_h`	Host name of the OID server.
`-ldap_p`	Port number of the OID server.
`-ldap_d`	Administration DN.
`-ldap_w`	Password for DN.
`-sso_h`	HTTP server Hostname for the SSO.
`-sso_p`	HTTP server port number for the SSO.
`-oh`	Oracle Home.
`-mc`	Middle-tier Installation.
	This includes the Oracle9*i*AS Portal's OID and Oracle9*i*AS Single Sign-On Server integration.
	Default value: TRUE
`-mi`	Midtier Configuration.
	This includes Oracle9*i*AS Portal's OC4J deployment and Oracle Enterprise Manager configuration.
	Default value: FALSE

3.2.7 SYSOBJECTS OPCA Mode

Action

Installs Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server required SYS schema dependencies as follows:

1. Installs PL/SQL Web Toolkit (OWA) packages
2. Installs VPD Context packages

Note:

- This mode needs to be run only once per database.
- The -sys option in the modes ALL, PORTAL and SSO provide the same functionality.
- Only a silent install is supported for this mode.

Usage example

Full Usage

```
ptlasst.csh -mode SYSOBJECTS -c myhost.domain.com:1521:mySID -p change_on_
install -silent -verbose
```

Table 3–8 List of supported parameters for the SYSOBJECTS mode

Parameter	Description
`-mode`	**Mandatory** install mode. The different install modes are PORTAL, SSO, SSOPARTNERCONFIG, LANGUAGE, MIDTIER, ALL, SYSOBJECTS.
`-c`	**Mandatory** connect string to the target database. The format should be *hostname:port:sid*.
`-p`	**Mandatory** SYS password for the target database.
`-silent`	Runs the OPCA in silent mode. Default value: TRUE

Table 3–8 List of supported parameters for the SYSOBJECTS mode

Parameter	Description
`-verbose`	This parameter enables the Oracle9*i* Reports integration with Oracle9*i*AS Portal. If this parameter is not set, logging information will be brief and the OPCA aborts the installation if it encounters any kind of ORA-,PLS- or SP2 errors. Default value: TRUE

4

Using the PL/SQL HTTP Adapter

This chapter provides information about the PL/SQL HTTP Adapter and on how to use it to share portlets with other Oracle9*i*AS Portal instances. Specific topics in this chapter covered include:

- About the PL/SQL HTTP Adapter
- Setting up the Environment to use the PL/SQL HTTP Adapter
- Registering A Provider Using The PL/SQL HTTP Adapter
- Writing Custom Portlets using The PL/SQL Http Adapter

4.1 About the PL/SQL HTTP Adapter

In this section we will describe the following:

- Overview
- Differences between Database Providers and Web Providers
- Use of the PL/SQL HTTP Adapter
- Security Issues
- PL/SQL HTTP Adapter related Portlet modifications

4.1.1 Overview

The PL/SQL HTTP Adapter is a component of Oracle9*i*AS Portal which allows Oracle9*i*AS Portal instances to share their database portlets via the web portlet interface. It is a new tool that uses SOAP and HTTP to distribute database providers across database servers. The PL/SQL HTTP Adapter allows database providers to be accessed as though they were web providers.

In earlier versions of Oracle Portal all database providers accessed from a portal instance had to be on the same physical database server that contained the portal instance.

In Oracle Portal version 3.0.9 it was possible to distribute database portlets across database servers. To do this the user had to register each portal 'node' with each other which created a database link between the 'nodes'. These portal nodes would not function beyond a firewall. Furthermore the registration of the portal nodes was symmetric, which made the registration of multiple nodes hard to manage

Portal already had the concept of *web providers* where the communication between the portal and the provider is done with the open protocols HTTP and SOAP. The PDK-Java services allow users to easily develop providers in Java that receive SOAP messages and respond accordingly.

The PL/SQL HTTP Adapter is a module written in the portal instance (in both Java & PL/SQL) that receives the SOAP messages for a web provider, parses the SOAP and then dispatches the messages to a database provider as PL/SQL procedure calls. In effect, the PL/SQL HTTP Adapter makes a database provider behave exactly the same way as a web provider. This allows users to distribute their database providers across database servers. All remote providers can now be treated as web providers, hiding their implementation from the user and effectively replacing the distributed Portal installations.

4.1.2 Differences between Database Providers and Web Providers

The biggest difference between database providers and web providers is that typically database providers use a portal session within the code, so that as part of the PL/SQL HTTP Adapter a portal session is created on the remote portal instance. The SOAP messages were extended to contain enough information to create a session on the remote portal instance, which means that the user in the remote portal must be the same user as in the local portal. For example, if 'UserA' is running in 'PortalA' and is using a provider on 'PortalB' via the PL/SQL HTTP Adapter then a session will be created in 'PortalB' for 'UserA'. Typically this means that 'PortalA' and 'PortalB' would share the same as partner applications. However an alternative arrangement could be that they have separate Oracle9*i*AS Single Sign-On Servers but the Oracle9*i*AS Single Sign-On Servers share the same name server. An example could be two Oracle9*i*AS Single Sign-On Servers sharing the same OID instance.

4.1.3 Use of the PL/SQL HTTP Adapter

The use of the PL/SQL HTTP Adapter can be divided into three categories:

Table 4–1 Use of the PL/SQL HTTP Adapter

Parameter	Description
Oracle9*i*AS Portal Database Providers	Portal Database Providers created within Oracle9*i*AS Portal will have the necessary code to be run through the PL/SQL HTTP Adapter. This means that applications created containing forms, charts, reports etc. can be shown on any other portal instance.
Pages	Pages exposed as portlets can also be run through the PL/SQL HTTP Adapter. Regions within pages can contain portlets or items. Using the PL/SQL HTTP Adapter these can now be accessed from any portal instance.
User Created Providers	Users may wish to create their own PL/SQL providers. You will be able to expose these providers through the PL/SQL HTTP Adapter as long as they are coded in accordance with the guidelines given in this chapter.

4.1.4 Security Issues

The PL/SQL HTTP Adapter creates a portal session in the remote portal based on the information passed in a SOAP message. This introduces a security issue since it may be possible to replicate these SOAP messages and create sessions for a any user on a portal and then access the portal as that user. To avoid this an encryption key is shared between the two portals and part of the SOAP message is encrypted using that key. The requested private portal session can only be created if it can be decrypted by the previously shared key. Otherwise a PUBLIC session is created. Show messages do not send SOAP and are protected by the encrypted cookie, which is created by the initSession SOAP message. Using this method the PL/SQL HTTP Adapter can safely trust the incoming SOAP message and create portal sessions in the portal instance without opening the portal to hackers.

See Also: Section 4.2.2, "PL/SQL HTTP Adapter User Authentication using HMAC"

If it is known that the portal instance will only be accessed via the PL/SQL HTTP Adapter from other portal instances then security can be enhanced by configuring the listener to restrict access from machines other than the known portal instances. This is done by using the 'Allow' directive in the httpd.conf file.

4.1.5 PL/SQL HTTP Adapter related Portlet modifications

It should be noted that database providers written before Oracle9*i* Application Server will not work when accessed via the PL/SQL HTTP Adapter if one of the following conditions is true:

- The portlet contains relative links.
- The portlet is customizable.

All links within a portlet should be absolute links, i.e. 'http://host:port/images/foo.gif' rather than relative, '/images/foo.gif' when using the PL/SQL HTTP Adapter. This is because the request is processed by the *Parallel Page Engin*e on the local portal instance. Relative links will therefore be interpreted as relative to the local portal and not to the portal containing the portlet.

Customization is an issue because the processing of customization is different between database and web providers. For web providers the customization form is submitted to the *Parallel Page Engin*e of the local portal, which in turn calls the portlet again and the customizations are saved and the page is redirected appropriately. Since database providers accessed via the PL/SQL HTTP Adapter are effectively web providers then this method of customization should be undertaken for these providers. A public API is provided (WWPRO_API_ADAPTER) to do this.

Portal Database Portlet Providers developed in previous versions of Oracle9*i*AS Portal will be upgraded automatically to work with the PL/SQL HTTP Adapter. Pages exposed as providers can also be accessed via the PL/SQL HTTP Adapter.

4.2 Setting up the Environment to use the PL/SQL HTTP Adapter

To use the PL/SQL HTTP Adapter there are a few administrative steps that must be undertaken. These steps are:

- Updating the DAD Name
- PL/SQL HTTP Adapter User Authentication using HMAC
- Setting the Cookie Domain
- Sharing an Oracle9iAS Single Sign-On Server and an OID Server

4.2.1 Updating the DAD Name

All portals that are accessed via the PL/SQL HTTP Adapter must have different DAD names, and the cookie names must match the DAD names. If this is not the

case, for example, if more than one portal has the default DAD name '*portal*', then a new DAD entry has to be added to the *dads.conf* file. This file is located under:

```
ORACLE_HOME/Apache/modplsql/conf/dads.conf
```

A typical entry in this file looks like this :

```
<Location /pls/portal>
    SetHandler pls_handler
    Order allow,deny
    Allow from All
    AllowOverride None
    PlsqlDatabaseUsername portal
    PlsqlDatabasePassword SomePassword
    PlsqlDatabaseConnectString myhost.domain.com:1521:mySID
    PlsqlDefaultPage portal.home
    PlsqlAuthenticationMode SingleSignOn
    PlsqlSessionCookieName portal
    PlsqlMaxRequestsPerSession 500
    PlsqlDocumentTablename portal.wwdoc_document
    PlsqlDocumentPath docs
    PlsqlDocumentProcedure portal.wwdoc_process.process_download
    PlsqlPathAlias url
    PlsqlPathAliasProcedure portal.wwpth_api_alias.process_download
    PlsqlFetchBufferSize 128
</Location>
```

To create a new DAD entry, make a copy of an existing entry, and change the value of *Location* to, for example, `/pls/portal2` and change the value for *PlsqlSessionCookieName* to `portal2`. The Location, or DAD name and the PlsqlSessionCookieName (the cookie name) must be the same. After saving the file, stop and restart the middle-tier using opmnctl.

After this, the Oracle9*i*AS Portal Configuration Assistant (OPCA) will have to be run in the MIDTIER mode. The example in Section 4.2.4, "Sharing an Oracle9iAS Single Sign-On Server and an OID Server" demonstrates how to update the DAD.

If, for example, the original schema and dad name was *portal* and the new dad name is *portal2*, the URL to access the adapter on the portal would be:

```
http://myApache.domain.com/adapter/portal2/portal
```

See also:

- Section 3.2.2, "MIDTIER OPCA Mode"
- Section 4.2.4, "Sharing an Oracle9iAS Single Sign-On Server and an OID Server"

4.2.2 PL/SQL HTTP Adapter User Authentication using HMAC

PL/SQL HTTP Adapter functionality will support the registering of remote Database providers between geographically dispersed portals. Database providers are registered as if they were web providers residing at a special URL on the remote portal.

In order that more than just public content can be rendered in the remote portlets we require that in some way we can guarantee that user A on one portal is the same as user A on another portal. This will typically be achieved by a shared Oracle9*i*AS Single Sign-On Server using the *Partner Application* feature, but may also be achieved with a shared name server (e.g. OID), synchronized name servers or a manual process.

If this environment can be achieved, then using the Hash Message Authentication Code (HMAC) authentication mechanism, private sessions can be initiated on a remote portal to render private content of remote portlets.

4.2.2.1 Setting the HMAC keys

If the administrator of portal A wishes to permit users of portal B to create private sessions on portal A, a private 'key' will have to be stored on each portal. This key is used to encode and decode portions of each SOAP request sent between them. If a key is missing or they are different on each portal, only PUBLIC sessions will be created.

A key must be at least 10 characters long, and the one administrator should inform the other administrator of its value in a suitably secure way.

SQL scripts are provided to perform the task of maintaining the key store - all are found in the *wwc* directory

Table 4–2 SQL Scripts for maintaining the key store

Script	Description
proadsss.sql	Sets the key at the sending end.
proadssr.sql	Sets the key at the receiving end.

Table 4–2 SQL Scripts for maintaining the key store

Script	Description
proadsds.sql	Removes the key at the sending end.
proadsdr.sql	Removes the key at the receiving end.

In each case 'sending' and 'receiving' refer to the SOAP message.

Example 4–1 Setting the HMAC Keys:

In the example mentioned above, portal B is the sender (sending SOAP and show requests) and portal A is the receiver of those requests. The portal Administrator of portal B must connect to SQL*Plus as the portal owner and run:

```
SQL> @proadsss
Enter provider portal PL/SQL Adapter URL:
http://<portalA_hostname>:<port>/adapter/<portalA_DAD>
Enter shared key:<shared key>
exit;
```

The portal Administrator of portal A must connect to SQL*Plus as the portal owner and run:

```
SQL> @proadssr
Enter provider portal PL/SQL Adapter URL:
http://<portalB_hostname>:<port>/adapter/<portalB_DAD>
Enter shared key:<shared key>
exit;
```

If sharing of providers is required both ways, then this will need to be repeated the other way round, possibly with different shared keys. It should also be noted that a portal can expose its providers to several other portal instances (e.g. 'Portal A' exposes providers to 'Portal B' and 'Portal C') and separate keys can be set up between each of the portal instances.

4.2.3 Setting the Cookie Domain

Normally cookie domains are restricted to a single machine. This can be widened by running a script on each portal, and then checking the '*Web provider in same cookie domain as the portal* ' box on provider registration. Once this is done '*deep link*' functionality can be achieved.

Cookies received by a browser, or other HTTP client, are sent to servers if the domain of the cookie matches the server's host name. So cookies with the domain '.co.uk' and 'mycompany.co.uk' will be sent with a request to 'http://mycompany.co.uk/pls/etc/etc'. By default the scope of cookies created by portal is restricted to the host name of the middle tier machine.

Since communication to the portlets is done in the middle tier by the *Parallel Page Engine* (PPE) and not the browser, the session cookie for the remote portal will, by default, not be sent to the remote portal when links are followed within the portlet.

This can be solved by widening the scope of the cookies created by portal and making sure that the cookies received by the PPE are sent back to the browser Widening the scope of the cookies created by portal is achieved by running the SQL script ctxckupd.sql in the wwc directory.

For example, there are two portals:

- http://myhost1.mycompany.co.uk:3000/pls/portalA
- http://myhost2.mycompany.co.uk:4000/pls/portalB

and a provider is registered from 'Portal B' onto 'Portal A'.

When showing a page on 'Portal A' that contained a portlet from 'Portal B' by default a portal session cookie for 'Portal B' whose domain is 'myhost2.mycompany.co.uk:4000' would be created, and sent to the PPE. If the '*Web provider in same cookie domain as the portal*' property is checked on the provider registration page then this cookie will be sent back to the browser, but the domain of the cookie will then be 'myhost1.mycompany.co.uk:3000' because that is where it is being sent from, because the PPE is at 'myhost1.mycompany.co.uk:3000').

If a link is followed from within the portlet the cookie is not sent with the request, because the domain of the cookie does not match with that of the host of the request.

To solve this, connect to SQL*Plus as the portal owner of each portal and run *wwc/ctxckupd.sql* and broaden the scope of the domain's cookies created by Oracle9*i*AS Portal so each portal is in the same domain. Once this is done the scope of the cookie domains created by any of the portals will be broad enough to be sent back to the browser. Links within the portlet will then work correctly.

> **See Also:** Section B.7, "Modifying the Scope of the Portal Session Cookie"

4.2.4 Sharing an Oracle9*i*AS Single Sign-On Server and an OID Server

Portal session information is passed to the remote portal which uses the PL/SQL HTTP Adapter to create a session. All portals you want to create private sessions on must share the same OID server and the same Oracle9iAS Single Sign-On Server.

For example if a user 'JSMITH' displays a page on one portal and a portlet on that page is being sourced from the PL/SQL HTTP Adapter on a remote portal, then a session is created on the remote portal for user 'JSMITH'. If the two portals do not share a Oracle9*i*AS Single Sign-On Server then 'JSMITH' may be the username for 'John Smith' on one portal and 'Jane Smith' on the other. Typically, this is solved by sharing a Oracle9*i*AS Single Sign-On Server between the two portal instances (i.e. they are partner applications). If the portlets being shown are 'public' then there is no need to share the Oracle9*i*AS Single Sign-On Server and a public portal session will be created at the remote portal instance.

In order to set up Oracle9*i*AS Single Sign-On Server sharing you have to run OPCA in the MIDTIER mode as shown in the following example, which updates the DAD name to portal2, sets the SSO Server and the OID Server.

Note: You run OPCA by invoking the ptlasst.bat, or ptlasst.csh script. Refer to Section 3.1.2, "Using OPCA in Stand-alone mode" for information on how to use OPCA in Stand-alone mode.

Example 4–2 OPCA MIDTIER mode for sharing a Single Sign-On Server

```
ptlasst.csh -i custom -mode MIDTIER -s portal -sp portal -c
myhost.domain.com:1521:mySID -sdad portal2 -o orasso -op orasso -odad orasso
-host myApache.domain.com -port 7777 -sso_c myhost.domain.com:1521:mySID
-sso_h myApache.domain.com -sso_p 7777 -pa orasso_pa -pap orasso_pa -ps
orasso_ps -pp orasso_ps -pd myPwdDBLink -p_tns orasso_ps -s_tns portal
-iasname myIAS -ldap_h myOID.domain.com -ldap_p 4032 -ldap_d cn=orcladmin
-ldap_w welcome1 -pwd welcome1 -verbose
```

See Also: Section 3.2.2, "MIDTIER OPCA Mode"

4.3 Registering A Provider Using The PL/SQL HTTP Adapter

Registering a provider through the PL/SQL HTTP Adapter is like registering any web provider. You must perform the following steps:

1. On the first page of the 'Register Provider' screen enter the 'Name', 'Display Name', 'Timeout' and 'Timeout Message' as you would normally. Make sure the 'Implementation Style' is set to 'Web'. Although the provider is actually written in PL/SQL all communication to it is as a web provider and not a database provider so it is important to set the 'Implementation Style' to 'Web'.
2. On the second page enter the URL of the adapter service. The syntax for the url should be:

   `http://host:port/adapter/dad/schema`

   If the dad and the schema are the same you can just use:

   `http://host:port/adapter/dad`

   where the host, port, dad and schema locate the remote portal instance. You can verify that this is the correct URL by pasting it into a browser.

   If the URL is correct you should get to a page with the message "Congratulations - you got to the adapter test page"
3. Check the 'Web provider in same cookie domain as the portal' check box. This will ensure that cookies generated from the provider will be sent back to the browser. Note that it may be necessary to broaden the scope of the cookies created by portal as described above.
4. Enter the 'Service Id'. This should be in the form 'urn:<provider name>'. Where <provider name> is the name of the provider on the remote portal instance, this is case sensitive and will be upper case. This is the information that the PL/SQL HTTP Adapter uses to locate the specific provider at the remote portal.

   Note that for page groups exposed as providers the name of the provider will be something like 'MYPAGE970D272EBE9D2D0FE034080020F7DA4B' it is important that you specify this 'Name' rather than the 'Display Name'.
5. In the 'User/Session Information' section select the 'User' radio button and set the 'Login Frequency' to be 'Once Per User Session'. These settings make sure that information is sent with the request to allow a portal session to be created on the remote portal instance.

4.4 Writing Custom Portlets using The PL/SQL Http Adapter

There are two main areas of code that need special attention when writing database providers that are accessed through the PL/SQL HTTP Adapter. They are:

- Relative Links

- Customization

4.4.1 Relative Links

Any links within portlets that are accessed through the PL/SQL HTTP Adapter should absolute rather than relative. If links are relative then they will not work since they will be relative to the local middle tier rather than the remote middle tier. For example, links should be of the form 'http://myhost.mycompany/etc/etc' rather than '/etc/etc'.

4.4.2 Customization

The way customizations work when accessing portlets through the PL/SQL HTTP Adapter is now very similar to the method used by JPDK portlets. There are two main areas of the portlet code that need to be changed to make customization work through the PL/SQL HTTP Adapter:

- The show call of the portlet needs additional logic to show the portlet in *edit_defaults* mode, or, if the parameter '*p_mode*' is null, in *customize* mode. If the '*p_mode*' is 'OK', 'APPLY' or 'RESET', then the customizations should be saved as appropriate.
- The <FORM> HTML tags generated for the customize page should be created using the procedure *wwpro_api_adapter.open_form*. This will ensure that the action for the form is correct, and that the correct parameters are passed upon page submission. The sequence of events when submitting the customization form is:

  1. The page submits to the 'local' PPE. There are several standard parameters that need to be sent with this submission (e.g. *_providerid*, *_dad*, *p_action*, etc.) as well as the parameters that are being customized. The procedure *wwpro_api_adapter.open_form* is supplied to make the generation of this submission as simple as possible.
  2. The PPE then shows the customization page again. However the '*p_action*' parameter will now be set so that during the *show_portlet* call of the portlet it will be one of the following settings:

     '**OK**' - In this case the customizations should be saved and then there should be a re-direct to the page containing the portlets.

     '**APPLY**' - In this case the customizations should be saved and the customization page is shown.

'**RESET**' - In this case the default values for parameters are queried and the customization page is shown.

5

Deploying Web Portals

A Web *portal* provides an organized, personal view of business information, Web content, and applications drawn from a variety of sources, including those outside the portal itself. Oracle9*i*AS Portal is designed to help you create Web portals that enable users to manage, access, and interact with this information, all from a common entry point. You can build any size Web portal, from a small departmental site, to a company-wide intranet, all the way to an Internet deployment that allows enterprises to share information with thousands of subscribers.

After your development team has built your Web portal, the next step is to deploy a production version of it. Successful deployment means that end users are able to access contents in a timely manner, without delays, errors, or server downtime. Because Oracle9*i*AS Portal can be installed in a variety of configurations on different machines, a successful deployment ultimately depends how you configure Portal to address the requirements of your site.

While designing your configuration, you should identify goals that you want your system to achieve, for example:

- Performance. Response time is the time between the receipt of a user request and the completion of the response to the request. Your Web site should respond as quickly as possible using the least amount of software and hardware overhead.
- Capacity. Servers, databases, and resources supporting your Web portal must handle wide variations in user traffic, especially during peak intervals.
- Scalability. If you expect your Web portal and its audience to grow over time, you need to be able to add more capacity.
- Availability. Most users expect a Web portal to be available 24 hours a day. Your configuration should be able to handle problems that cause one of its components to fail, with minimal down-time.

- Security. Sensitive data should be secure without affecting content that you want to make available to all users

This chapter provides information on various deployment configurations. It contains the following sections:

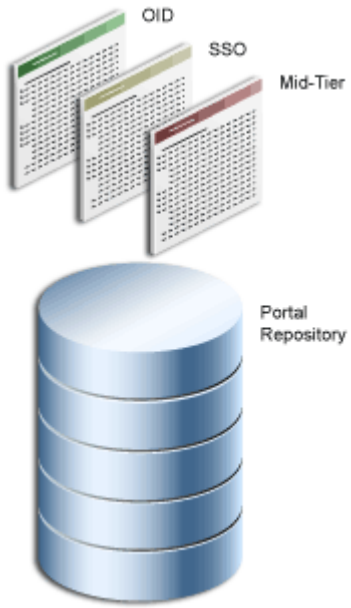
5.1 Oracle9*i*AS Portal architecture overview

As shown in Figure 5–1 on page 5-3, an Oracle9*i*AS Portal installation contains these components:

- Oracle9*i*AS Portal enables you to build, manage, and deploy the pages, portlets and applications that will be included in your Web portal.
- Oracle9*i* Application Server middle-tier provides several component pieces, including:
  - Oracle HTTP Server - an HTTP listener that handles all incoming HTTP requests to the Web portal by forwarding them to the appropriate Portal database provider or Web provider.
  - Parallel Page engine (PPE)- a servlet that assembles Portal pages, executes the portlets on them, and reads and writes content to Oracle9*i*AS Portal and Oracle9*i*AS Web Cache.
  - mod_plsql - executes PL/SQL procedures residing in the Oracle9*i* database - for example, the portlets that you create using Oracle9*i*AS Portal - and generates HTTP responses.
  - Oracle9*i*AS Web Cache - works with Oracle9*i*AS Portal's own file-based caching to cache page definitions and content in memory.

- **Oracle Internet Directory (OID)** - an LDAP repository for storing user credentials and group memberships for Oracle9*i*AS Portal and other Oracle products.
- **Single Sign-On Server (SSO**) - authenticates user credentials against OID for Oracle9*i*AS Portal and other applications, thus enabling users to log on to the Web portal a single time to access multiple accounts and applications with a single username and password.
- **Oracle9*i* database** -stores the objects that comprise Oracle9*i*AS Portal, OID and SSO. The part of the database where they are stored is known as the Oracle9*i*AS Portal Repository.

Figure 5–1 Oracle9iAS Portal architecture



You can install multiple instances of any of these components on multiple servers, then connect the servers to suit your needs. Deployment configuration options for Oracle9*i*AS Portal range from installing everything on a single machine to multi-tier configurations in which the pieces comprising Oracle9*i*AS Portal are located across multiple servers. Before you begin, it's important that you understand how Oracle9*i*AS Portal components work together.

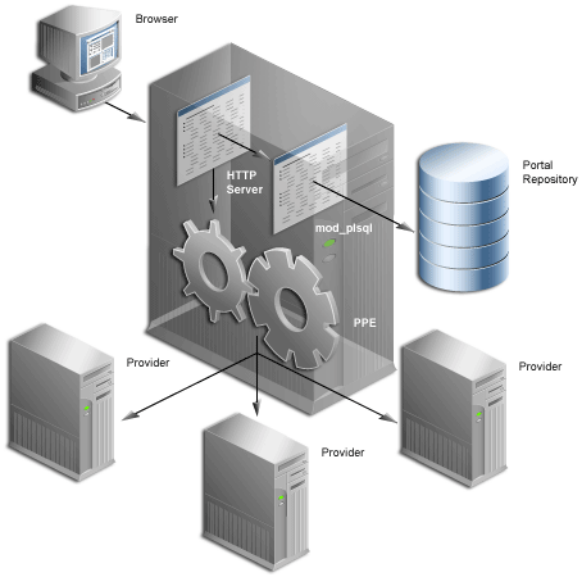
See also: *Oracle9i*AS Portal *Architecture Overview* located on Oracle Technology Network (http://portalcenter.oracle.com)

5.1.1 Assembling Oracle9*i*AS Portal pages

Each time a user requests an Oracle9*i*AS Portal page, the page is dynamically assembled and formatted according to the portlets and layout chosen for that page. Keep in mind that the parts that comprise the page are typically drawn from a variety of sources. For example, the page's layout, look and feel, and user customizations are stored in the database as part of the overall page definition, completely separate from any page content. This information may, in turn, be cached by the middle-tier.

The HTML portlets that appear on the page can be written in PL/SQL or Java. If PL/SQL, the source of the portlet is the Oracle9*i*AS Portal database. This could be the database where the current instance of Oracle9*i*AS Portal is installed, or some other Portal database located on a remote server. If written in Java, a Web provider provides the portlet from some location accessible through the Internet. For example, you could create a Portal page that displays portlet content from an external Web provider and another portlet, containing a chart, that you created using Oracle9*i*AS Portal.

Figure 5–2 HTTP Server handles Web requests for Portal pages by forwarding them to mod_plsql or the PPE



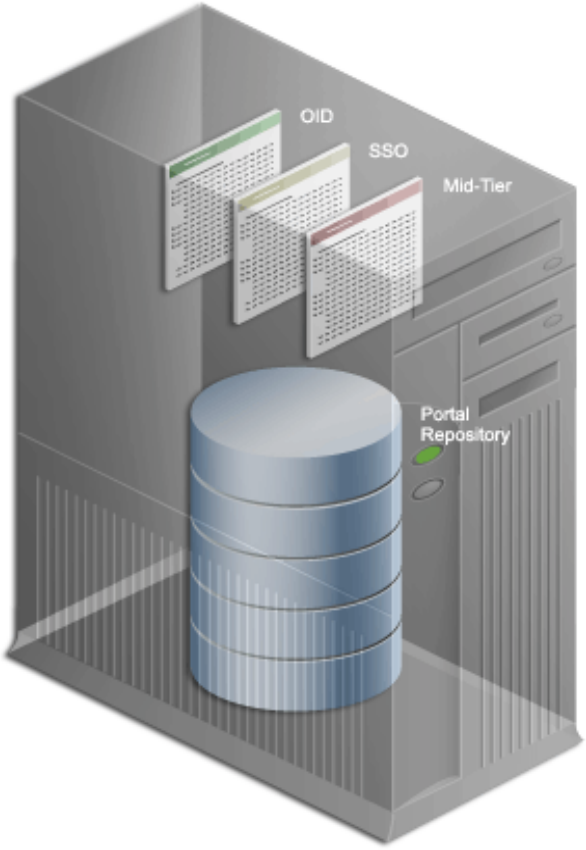
As shown in Figure 5–2, when a user requests a Portal page, the request is forwarded from the user's browser to Oracle HTTP Server, the middle-tier listener. If the incoming request is to display a Portal page, the listener hands the request off to the Parallel Page Engine (PPE). The PPE then sends requests, in parallel, to the page's various portlet providers to execute the portlets and then assembles the page from the resulting portlet content.

If the request is to execute a PL/SQL procedure, the listener hands it off to mod_plsql, which forwards it to the Oracle9*i*AS Portal database.

5.2 Single machine configuration

In the simplest configuration, all of the component pieces described in "Oracle9iAS Portal architecture overview" on page 5-2 could be set up on a single machine. A single database could reside on the machine, containing separate schemas for Oracle9*i*AS Portal, OID, and SSO as shown in Figure 5–3.

Figure 5–3 Oracle9iAS Portal single machine configuration



This configuration would work nicely in a small development environment in which your developers are using Oracle9*i*AS Portal's declarative interface to build pages, portlets and applications. It could also easily support a small deployment of the finished Web portal.

If you expect to deploy a larger site that delivers more content to more users, you'll need more than a single server or the simple configuration shown in Figure 5–3.

5.3 How much hardware and software?

As with any Web portal, the server and database capacity you'll need to deploy a portal built using Oracle9*i*AS Portal largely depends on the number of user requests

for pages that you anticipate. Displaying a single page to a user may require many separate transactions, from verifying whether the user has permission to view the page, to loading the images that appear on the page, to calling a style sheet that contains formatting information for the page.

The upper and lower limits of what you'll need are determined by how you expect your users to use the portal. At a minimum, you'll need enough server capacity to satisfy the average load during a work day, with response times that are acceptable to your user base. If possible, you should strive to satisfy the volume of page requests you anticipate during peak intervals of high user activity. Hardware resources such as CPU, memory, I/O capacity, and network bandwidth are key to reducing response times. Unless you install Oracle9*i*AS Portal on a server or group of servers that can handle a large number of transactions, your users are probably going to experience slow response times.

The same is true of your database. If you have many applications competing for the same database resources, your Web portal performance may suffer. You can install multiple instances of Oracle9*i*AS Portal in the same database, for example, a production instance for developing new pages and portlets, and a separate instance for deploying your finished Web Portal. You need to consider whether your database can satisfy requests from both instances in a timely manner.

Adding more servers and database capacity will certainly improve your Web portal's performance, but unless you have unlimited funds at your disposal, you'll need to balance good performance against the costs associated with each new piece of hardware and software.

5.4 Deploying larger Web portals

If a single machine configuration will not suit your needs, you should consider moving the various pieces of the Oracle9*i*AS Portal architecture to other machines. A rule of thumb when configuring your Web portal is: the larger the site, the more servers you'll require, each server performing more specialized work. Adding extra hardware increases performance. Adding more software instances supports *redundancy*.

Deployment options for configuring larger Web portal sites include:

- Installing the middle-tier separately from the database
- Installing SSO and OID separately from the middle-tier
- Installing SSO and OID separately from the database
- Adding redundant software instances

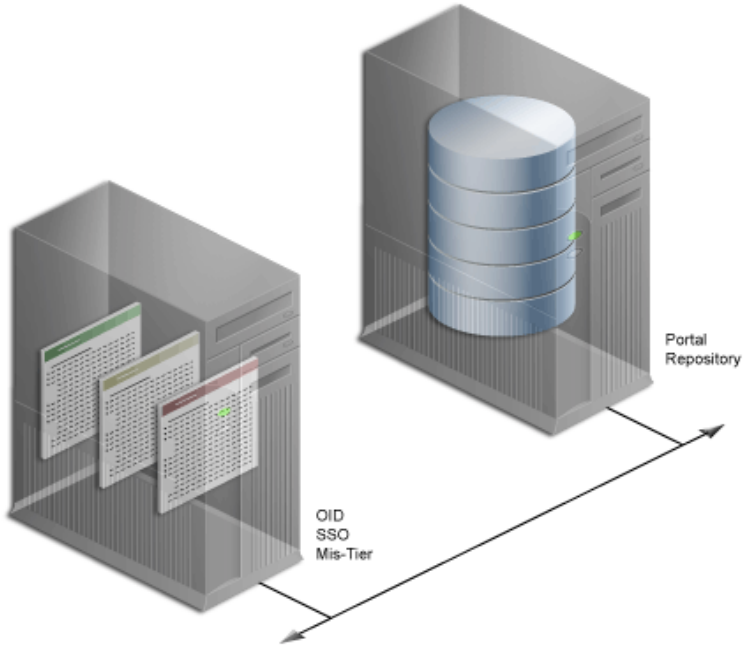
These tasks should be performed in the above order until you are satisfied your configuration will handle the demands of your deployed Web portal. If your site needs to handle only a moderate workload, for example, you should first separate the middle-tier from the database, then think about moving SSO to another server. You probably won't need to perform all of the above configuration tasks. But as the site grows, you should expand its underlying configuration by following the sequence shown in the list above.

Before you go on-line with your Web portal, it's a good idea to set up and test a small pilot system. This allows you to gather valuable configuration and tuning information based on real usage patterns, without affecting the users you eventually plan to serve.

5.4.1 Separating the middle-tier from the database

The first thing you should consider when configuring a larger system is installing the middle-tier separately, preferably on a different machine from the Oracle9i database as shown in Figure 5–4.

Figure 5–4 Deployment configuration with Oracle9iAS Portal and database on separate machines



This frees the database and middle-tier from having to compete for hardware resources such as such as I/O, memory, and disk space. Installing them on separate machines also gives you more flexibility in tuning performance. Tuning parameters, such as those for an operating system, are different than those for middle-tier components such as the HTTP server. Setting a performance parameter for one may not provide optional performance for another.

5.4.2 Installing Oracle9*i*AS Single Sign-On Server and OID separately

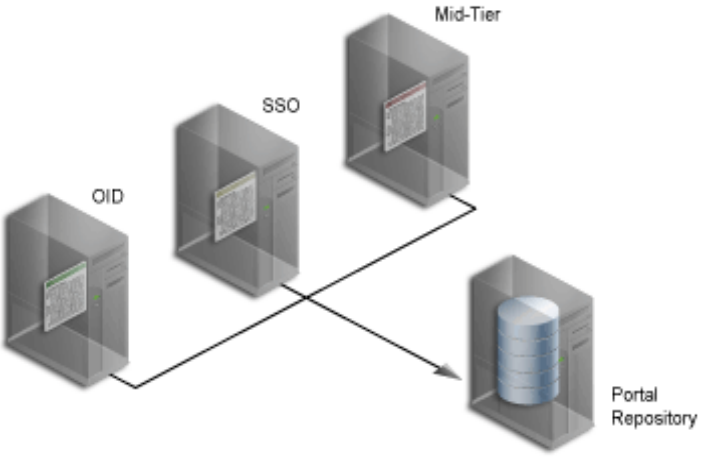
Oracle9*i*AS Single Sign-On Server enables Oracle9*i*AS Portal users to identify themselves to multiple applications by logging into a single page similar to the one shown in Figure 5–5.

Figure 5–5 Single Sign-on page

Once users have logged into a deployed Oracle9*i*AS Portal site, they can access these applications from portlets in the portal.

As shown in Figure 5–6, a single instance of SSO can be configured to work with multiple Oracle products, including multiple instances of the Oracle9*i*AS Portal middle-tier.

Figure 5–6 Deployment configuration with separate SSO and OID instances



The system shown in Figure 5–6 is an example of a *distributed configuration.* The configuration includes a centralized SSO server supporting multiple middle-tier instances. Moving SSO to its own server gives you the flexibility to tune its performance independently of the database and middle-tier.

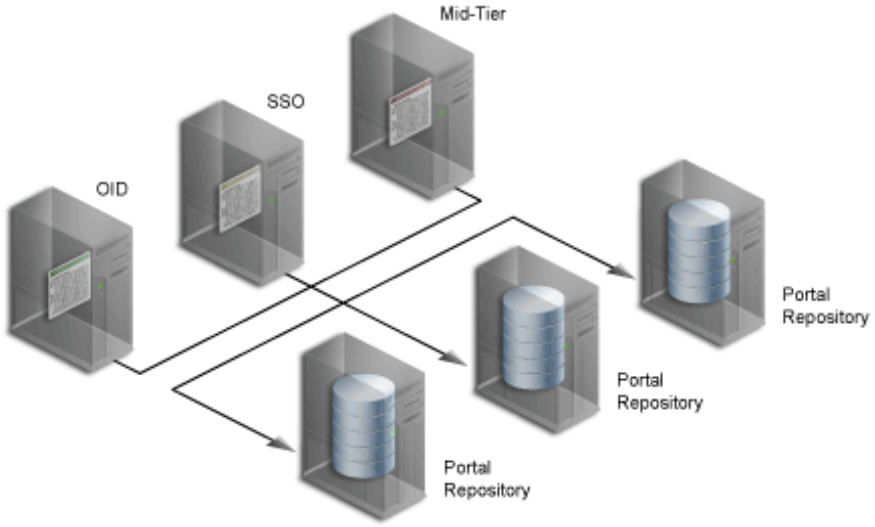
In addition, isolating SSO from middle-tier installations ensures greater stability for the entire distributed system. If the machine where a middle-tier is installed fails, the Single Sign-on Server and other middle-tier instances that rely on it to validate logins are not affected.

After you move SSO off to its own server, consider doing the same for OID. In a typical Oracle9*i*AS Portal configuration, OID works with SSO to validate user credentials. It also keeps track of memberships in Oracle9*i*AS Portal groups. Like SSO, it's a good idea to install, tune, and maintain OID on a dedicated server insulated from the rest of your configuration.

5.4.3 Providing databases for SSO and OID

As your Oracle9*i*AS Portal system grows, you may next want to provide separate databases for Oracle9*i*AS Portal, OID, and SSO, rather than provide each with a separate schema in the same database, as shown in Figure 5–7.

Figure 5–7 Deployment configuration with separate databases for SSO and OID



All databases could be on the same server if the machine is large enough. If not, consider installing each database on its own server in order to provide adequate hardware resources and enhance performance tuning.

5.4.4 Adding middle-tier instances

You can add redundant middle-tier instances, each with identical configuration settings, to support the largest Web portals. It's a good idea to install each middle-tier instance on its own machine to make failures more independent of one another.

The middle-tier forwards user requests for portal pages to a database or application provider, then assembles the pages with the returned content. As you add more middle-tier instances to your Oracle9*i*AS Portal configuration, you increase the number of user requests that can be forwarded and improve the overall performance of your portal. In addition, because the middle-tier performs some processing before forwarding a request, less time is spent sending and receiving data over the network. Database and network resources are used more efficiently.

5.5 Getting the most out of your configuration

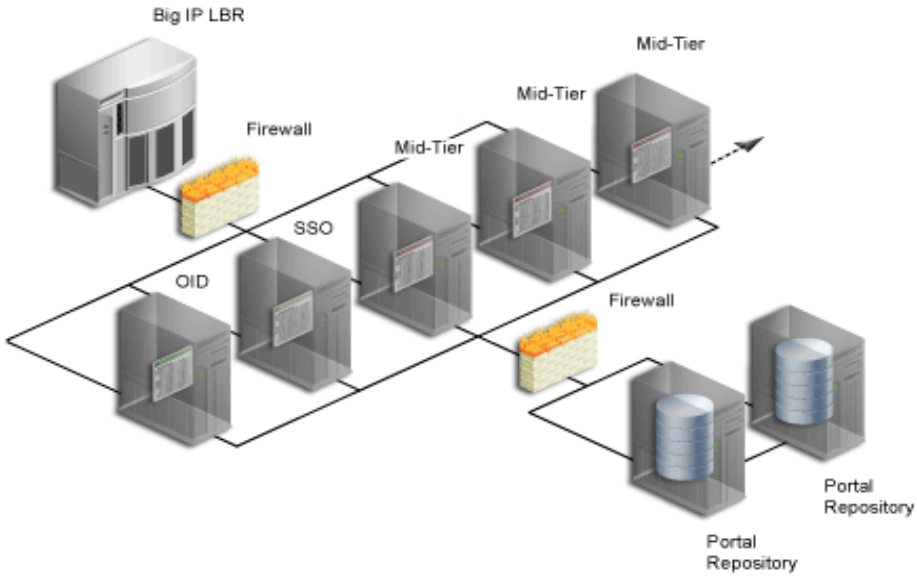
A distributed Oracle9*i*AS Portal configuration offers improved performance over a single machine configuration because you are making more software and hardware resources available to the Web portal. But there are other benefits. You can use additional servers and software to provide *failover*, thus ensuring system stability. And you can deal with wide fluctuations in the amount of work your Web portal is expected to perform over the course of a day using *load balancing* between multiple servers. Finally, you can add more servers to a distributed configuration in order to support more users, thus providing *scalability*.

5.5.1 Load balancing

If you anticipate a heavy volume of traffic on your Web portal, you can distribute the load across multiple servers, each with its own middle-tier instance. If one server is overloaded with too much traffic, a second server can handle the overflow.

Oracle9*i* Application Server provides its own load balancing capability by pooling server instances to service incoming requests. If one instance does not respond, then the request is forwarded to another instance. This ensures that content and applications are always available to users of your deployed site.

Figure 5–8 Multiple server configuration employing a Load Balancing Router



For very large sites, you can add a Load Balancing Router (LBR) to distribute incoming requests to the middle-tier servers, as shown in Figure 5–8. An LBR is a very fast network device that distributes Web requests to a large number of servers. It provides users of your Portal with a single published address, instead of them having to send each request to a particular middle-tier server.

My.Oracle.com (MOC) uses a BIG-IP router to sort requests. Because the software logic for distributing loads is contained in the LBR itself rather than installed separately on each individual middle-tier server, an LBR lowers the overall administrative costs of your configuration. MOC is both an intranet and extranet Web site. It provides Oracle customers and employees with a single customizable entry point to all of Oracle's on-line services as well business information from external providers such as NASDAQ and Business Week.

Adding an LBR can also help your configuration deal with load variations. Users may access your site, use its applications, and request content at a much higher frequency during certain peak intervals, for example, between 9 AM and 10 AM when most users log on to begin their work day. During these periods of heavy traffic, the LBR can distribute page requests among the various middle-tier instances to ensure quick response times.

If your peak load occurs on a regular basis, consider a configuration that specifically addresses the need to handle peak load requirements. If your peak load is infrequent, you may be willing to tolerate slower response times at peak intervals rather than spend additional money on hardware.

Note that the LBR itself can be configured to support failover. The My.Oracle.com configuration in Figure 5–8 could add a second BIG-IP router, which would be available in case the primary router fails.

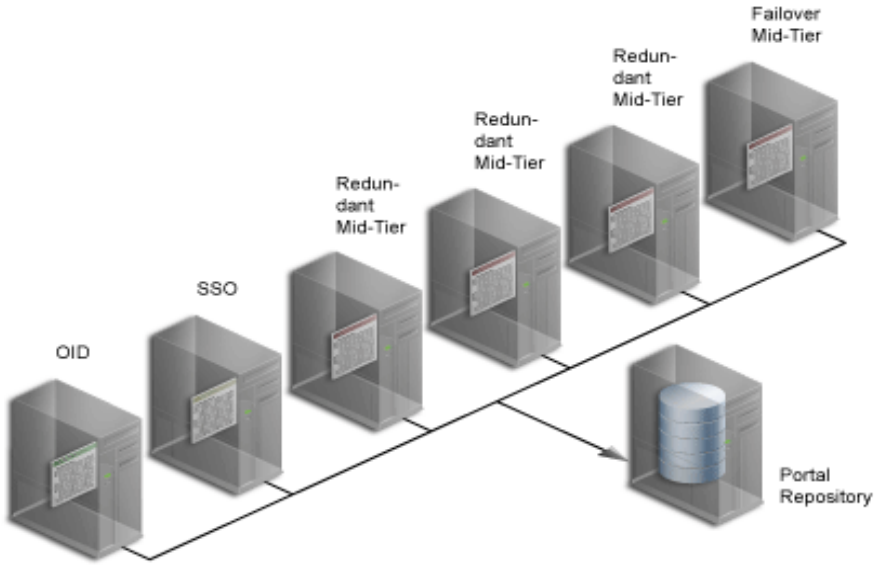
5.5.2 Failover

Failover is the ability to switch to backup when part of your system fails, such as a server or database. When an Oracle9*i* database fails, for example, it restarts using any preserved state information from the backup.

Redundancy is the technique of providing duplicate machines configured identically. The redundant machines provide enough capacity to service requests, and provide backups in case of failures and errors. You implement redundancy by increasing the number of machines in your configuration. One server is typically active while the other monitors the first server's activity, ready to take over if it fails.

As shown in Figure 5–9, My Oracle.com provides for failover using an additional middle-tier server that can take over if any of the other servers encounters problems that cause it to fail.

Figure 5–9 My Oracle.com middle-tier configuration with backup, inside the firewall. An LBR located outside the firewall distributes requests to the middle-tiers



To set up redundant middle-tier instances, you configure the original and each redundant instance with identical server name and server port entries, for example, *my.oracle.com* and port *5000*.

One alternative to redundancy is to failover by using any excess capacity that you have in your overall configuration. For example, you might have four middle-tier servers, each running at 75% capacity. If one server fails, the other three can take over the workload of the fourth (25% X 3 = 75% capacity of the failing server).

5.5.3 Scalability

Scalability is the ability of a Web portal to handle more requests as the number of users and the volume of content increases over time. As the portal handles more traffic, users should not notice any change in performance, as measured by response intervals and frequency of errors. If scalability is your goal, you need a flexible configuration that will allow you to add database capacity and servers incrementally as needed without adversely affecting the rest of your configuration.

When My.Oracle.com (MOC) was set up, for example, it was initially expected to serve approximately 40,000 Oracle employees. The user base is anticipated to

expand eventually to a million and a half, most of them users of the Oracle Technology Network (OTN), each automatically provided with an MOC account.

5.5.4 Caching

Oracle9*i*AS Portal uses three methods to cache Web pages and content.

- **Invalidation-based caching** is performed using Oracle9iAS Web Cache. An item remains in the cache until some event occurs that requires it to be refreshed. For example, a user may update some item, requiring the cache to be updated. In response to the event, the Oracle9iAS Portal Repository or a Provider sends an invalidation message to Oracle9iAS Web Cache. The next time there is a request for the invalidated item, it is refreshed in the cache.
- **Validation-based caching** is performed using the Oracle9iAS Portal Cache. Before an item in the Oracle9iAS Portal Cache is used, the Parallel Page Engine, or mod_plsql, contacts the Portal Repository or a Provider to determine if the cached item is still valid.
- **Expiry-based caching** also uses the Oracle9iAS Portal Cache. A retention period for the item specifies how long it is valid in the cache, before a refresh is required. Pages that use expiry-based caching may also be cached in the user's browser.

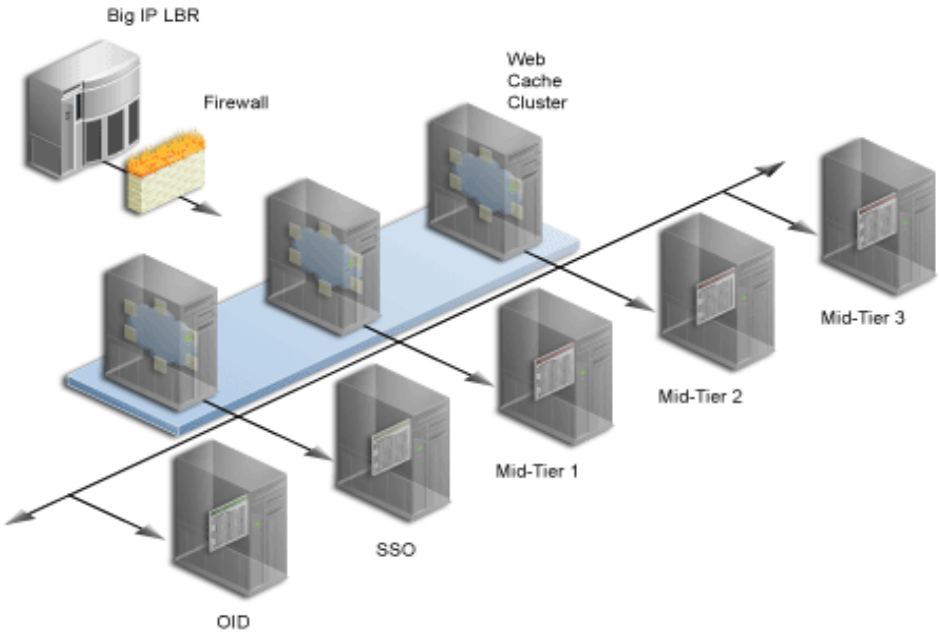
5.5.4.1 Configuring Oracle9*i*AS Web Cache

To increase the availability and scalability of medium to large deployments, consider configuring multiple instances of Oracle9*i*AS Web Cache to run as members of a cache cluster. A cluster is a collection of cooperating Oracle9*i*AS Web Cache instances that work together to provide a single logical cache. Cache clusters provide failure detection and failover, increasing the availability of your Web site. If an Oracle9*i*AS Web Cache instance fails, other members of the cache cluster detect the failure and take over ownership of the cached content of the failed cluster member.

By distributing the Web site's content across multiple Oracle9*i*AS Web Caches, more content can be cached and more client connections can be supported, expanding the capacity of your Web site. You make use of the processing power of more CPUs and, because multiple requests are executed in parallel, you increase the number of requests that are served concurrently

In addition to providing failover, an Oracle9*i*AS Web Cache cluster also balances the load it forwards to the middle-tier.

Figure 5–10 Adding Oracle9iAS Web Cache to a medium to large Portal configuration



In Figure 5–10 , the LBR distributes incoming requests to the three Oracle9*i*AS Web Cache instances. Each instance determines if the request matches on already in the cache. Because the instances in the cluster communicate with one another, all three instances are checked for the cached content. If there is a match, the cached content is returned to the Browser. If not, the request is forwarded to the middle-tier.

To take advantage of Oracle9*i*AS Web Cache's clustering capability, you must configure each instance as a member of a cache cluster. It is recommended not to use "sticky routing" in this configuration. Enabling sticky routing in the cluster would mean that requests from one Oracle9*i*AS Web Cache instance for a given user always needs to be sent to a particular middle-tier. This would prevent effective load balancing between the middle-tiers and also minimize the shareability of content in the cluster. In this setup there is no one-to-one relationship between an Oracle9*i*AS Web Cache instance and a matching middle-tier instance. As shown in Figure 5–10, Oracle9*i*AS Web Cache 1 provides load balancing between middle-tiers 1, 2, and 3. Oracle9*i*AS Web Cache 2 and 3 do the same.

To take advantage of Oracle9*i*AS Web Cache's load balancing capability, you must configure each instance as a members of a cache cluster. "Sticky routing" means that

certain types of requests are always forwarded from a Oracle9*i*AS Web Cache instance to middle-tier server. It is recommended not to use "Sticky routing" in this configuration, because in a clustered configuration, there is no one-to-one relationship between a Oracle9*i*AS Web Cache instance and a matching middle-tier instance. As shown in Figure 5–10, Oracle9*i*AS Web Cache 1 provides load balancing between middle-tiers 1, 2, and 3. Oracle9*i*AS Web Cache 2 and 3 do the same.

As a rule of thumb, one Oracle9*i*AS Web Cache instance can balance load across three middle-tier servers. If you need to support more middle-tiers, you can set up additional Oracle9*i*AS Web Cache instances. For example, Oracle9*i*AS Web Cache instance 1could provide load balancing to middle-tier servers 1 through 3. Oracle9*i*AS Web Cache instance 2 could provide it to middle-tier servers 2 through -4, instance 3 to middle-tiers 3 through 5, etc.

See also:

- *Oracle9i* Application Server *Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library.
- *Deploying Highly Scalable and Secure Portals with Oracle9i*AS Portal *Release 2,* a white paper located on the Oracle Technology Network at http://portalcenter.oracle.com

5.5.4.2 Configuring Oracle9*i*AS Portal Cache

If you have multiple middle-tiers in your configuration, you have the option of setting up the Oracle9*i*AS Portal Cache for each middle-tier on a shared file system. This ensures that each middle-tier can share cached content, rather than each drawing from its own independent cache.

For example, one middle-tier might handle a request for an item by caching it in the Oracle9*i*AS Portal cache. Because you typically use a load balancer for configurations having multiple middle-tiers, the next request for the item could be handled by a different middle-tier. This middle-tier could access the cached version if Oracle9*i*AS Portal Caches for each middle-tier are shared on a common file system.

5.6 Security

Oracle9*i*AS Portal can be configured to use Secure Sockets Layer (SSL), which provides security between browsers and servers. The Oracle9*i*AS Portal middle-tier as well as Oracle9*i*AS Single Sign-On Server can run in HTTPS mode.

See also:

- Section 2.8, "SSL Configuration"
- Section 2.10, "Configuring Oracle9iAS Portal Security"

5.6.1 Authenticating users

Depending on how you set up your configuration, you can use SSL to address the security needs of your Web portal as well as optimize its performance. For example, if you are mainly concerned with protecting user login credentials, consider running your Oracle9*i*AS Single Sign-On Server instance in HTTPS mode, which provides encryption and requires some overhead to support. Run instances of the middle-tier and Oracle9*i*AS Portal in HTTP mode.

5.6.2 Securing content

You may also want to encrypt the traffic between servers, and between servers and databases. For example, you might have a database that contains sensitive payroll data. In this case, you can run each middle-tier handling sensitive data in SSL mode.

One problem with this configuration is that you are supporting SSL on multiple servers. As the configuration shown in Figure 5–8 on page 5-13 indicates, you can set a Load Balancing Router outside your company's firewall while keeping your middle-tier servers inside. The LBR in this configuration will handle the Web traffic to and from your servers. LBRs such as the Big IP router that My.Oracle.com can add an SSL accelerator card that secures all data transmitted outside the firewall. An advantage of using this card is that it handles traffic much faster than would SSL software.

5.7 Tuning performance

After you set up the basic configuration of your Portal system, you are ready to tune performance based on the configuration. One way to optimize performance is to set the approximate number of simultaneous Web requests that your portal can handle. You can do this by setting options in the HTTP Server and Parallel Page Engine.

5.7.1 Setting the number of server processes

Oracle HTTP Server processes Web requests by distributing them to HTTP processes. HTTP Server can serve all types of requests originating in users' browsers, such as those for static files, Java servlets, or PL/SQL procedures.

MaxClients is an HTTP Server configuration directive that controls the maximum number of Web requests that the HTTP Server can handle at any given time. When the **MaxClients** value is exceeded, the HTTP Server refuses to handle any new requests until it handles the current load and the HTTP processes are freed. In fact, client browsers may be "locked out" if the number of allowable sessions has been exceeded by other browsers.

One way to think of the **MaxClients** directive is that it's a throttle that permits just the right flow of concurrent Web requests to your server. Set it too low, and your Web portal performance may suffer. Even though you may have the server and database resources to handle more traffic with quicker response intervals, Web requests can't get through because you haven't set enough processes in **MaxClients.**

Setting **MaxClients** too high unnecessarily consumes resources, because each http process server consumes resources such as CPU time, memory, and I/O. And it may result in poorer rather than better performance. Why? Keep in mind that the HTTP Server can handle all sorts of requests, including those for PL/SQL procedures. When the HTTP Server receives such a request, it hands it off to mod_plsql to communicate with the Portal database. For each request, mod_plsql opens a database connection. The value you set for **MaxClients**, therefore, sets the upper limit of database connections that mod_plsql can open.

Say you set **MaxClients** to the maximum number, 1024. At any given time, the HTTP Server is ready to handle 1024 simultaneous Web requests, including a number that require database connections. Even if your server is large enough to deal with this, the database to which it is connected may not be. And if the ratio of requests for PL/SQL procedures versus other types of requests suddenly becomes very high, you risk overloading your database.

The key to good performance is determining the number of Web requests the servers in your configuration can process as well as how much traffic your database can handle. So if your Portal configuration includes multiple middle-tier servers connected to a single database, the number of possible Web requests you can handle is probably going to be limited more by database capacity than the middle-tiers.

See also:

- *Oracle HTTP Server Administration Guide* in the Oracle9*i* Application Server documentation library.
- Section 6.7.1, "Configuring the MaxClient Setting"

5.7.2 Setting the number of idle processes

MinSpareServers is an HTTP Server directive that sets the minimum number of idle sessions. An idle sessions is one that is not currently handling a Web request. If the number of idle sessions is fewer than the number specified in **MinSpareServers**, new processes are created at a maximum rate of 1 per second.

You should consider tuning this parameter only on very busy sites. The default setting is 5. Setting this parameter to a large number is almost always a bad idea. A rule of thumb is to set **MinSpareServers** at a little over the average number of Web requests your Portal typically handles. Ideally, you can set it so user requests are filled all the time by open ports without having to open a new one, but this is possible if you have the database resources to support a lot of ports.

See also: *Oracle HTTP Server Administration Guide* in the Oracle9*i* Application Server documentation library.

5.7.3 Setting the number of PPE fetchers

As described in "Assembling Oracle9iAS Portal pages" on page 5-4, a request for a Portal page originates in the form of a URL sent from a user's browser to the HTTP server. If the request is for a Portal page, it is forwarded to Parallel Page Engine (PPE). The PPE then asks each Web provider that owns a portlet on the page to execute the portlet and return content to the Portal page.

The PPE uses a pool of *fetchers* to forward requests over the Internet to Web providers and wait for data to be returned. Once it is finished with the request, the fetcher is available to handle another new request.

Pool size is an optional PPE setting that determines the number of fetchers available at work at any given time. If the maximum is exceeded, new URL requests are placed in a queue waiting for a free PPE fetcher.

The default setting is 25. For most Web portals, you should never have to change pool size. But keep in mind that if pool size is too low, the user notices that pages take too long to draw at peak periods. If pool size is set too high, a possible resource

drain may occur because too many concurrent URL requests can overwhelm the PPE.

> **See also:** *Oracle HTTP Server Administration Guide* in the Oracle9*i* Application Server documentation library.

6

Advanced Oracle9*i*AS Portal Configuration

In the previous chapter we discussed various Oracle9*i*AS Portal configurations, ranging from a single machine configuration to a full scale enterprise deployment scenario, using Load Balancing routers, as well as multiple middle-tier servers and Oracle9*i*AS Web Cache clusters.

This chapter discusses the configuration that needs to be performed to accommodate some of these more advanced configurations. Specific topics include:

- Configuring Virtual Hosts
- Parallel Page Engine Configuration
- Configuring Reverse Proxy Servers Over the Internet
- Configuring Load Balancing Routers
- Setting up Two Sites Using the Same Infrastructure
- Configuring Multiple Middle-Tiers to Use the Same Infrastructure
- Tuning the Oracle HTTP Server
- Configuring Oracle9iAS Portal to Work with Oracle9iAS Web Cache

6.1 Configuring Virtual Hosts

The Oracle9*i* Application Server HTTP server supports the configuration of virtual hosts. This allows a single machine and port to represent a number of virtual hosts. To configure virtual hosts, you must set this up on both Oracle9*i*AS Portal as well as on the Oracle HTTP Server.

In our example, let's assume that we want to access Oracle9*i*AS Portal as `http://www.abc.com` as well as `http://www.xyz.com`. Also, let's assume that the Oracle9*i*AS Single Sign-On Server's URL is `http://www.login.com`.

To configure virtual hosts, open and edit the Oracle HTTP Server configuration file, `httpd.conf located in ORACLE_HOME/Apache/Apache/conf`. Verify that the contents of the file includes the similar information in the `Virtual Hosts` section:

```
### Section 3: Virtual Hosts
NameVirtualHost 127.0.0.1

<VirtualHost 127.0.0.1>
    ServerName www.abc.com
</VirtualHost>

<VirtualHost 127.0.0.1>
    ServerName www.xyz.com
</VirtualHost>

<VirtualHost 127.0.0.1>
    ServerName www.login.com
</VirtualHost>
```

See also: Section 2.9.3, "Configuring HTTPS with Virtual Hosts"

This example uses the IP address 127.0.0.1, which represents the local machine. This can be any valid IP address.

The domain names specified in the `ServerName` entries need to be valid domain names. If you are setting up Oracle9*i*AS Portal on a local laptop, make the appropriate entries in your local hosts file.

```
# Copyright (c) 1993-1995 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP
# for Windows NT/2000.
#
127.0.0.1 localhost
127.0.0.1 www.abc.com
127.0.0.1 www.xyz.com
127.0.0.1 www.login.com
```

For Single Sign-On on the Oracle9*i*AS Single Sign-On Server to work properly, it must always be referenced by any Partner Application with the same hostname in the URL. This is because cookies are sent back only to the host that generated them. So, in the preceding example, the Oracle9*i*AS Single Sign-On Server must always be

referenced as http://www.login.com. Thus, you must register www.abc.com, www.xyz.com, and www.login.com as Partner Applications:

1. Add a Partner Application entry for www.abc.com.
2. Add a Partner Application for www.xyz.com.
3. Add a Partner Application for the Oracle9*i*AS Single Sign-On Server, www.login.com, by running the Oracle9*i*AS Portal Configuration Assistant Configuration Assistant in the SSOPARTNERCONFIG mode.

   See also: Section 3.2.4, "SSOPARTNERCONFIG OPCA Mode" for SSOPARTNERCONFIG mode usage information.

6.2 Parallel Page Engine Configuration

The Oracle9*i*AS Portal architecture is designed around a three-tier architecture that allows any browser to connect to it. This flexible architecture allows each component (browser, Oracle HTTP Server listener, Oracle9*i* database, and Oracle9*i*AS Portal) to be upgraded individually as required.

6.2.1 Configuring Parallel Page Engine Parameters

When a page is requested from Oracle9*i*AS Portal, the request is made from the browser to the Oracle HTTP Server listener. The returned page is comprised of many types of portlets. A portlet is an area on a portal page that contains data from a particular data source.

The Parallel Page Engine obtains the page metadata from the Oracle9*i*AS Portal Repository and is responsible for assembling the portlets on the page.

6.2.1.1 Setting PPE Configuration Parameters

With the release of Oracle9*i* Application Server version 9.0.2 and above, all of the servlets are installed under OC4J, based upon the application deployment. All of the configuration parameters for PPE are entered in the *web.xml* file related to the PPE Deployment. In the default installation, this file can be found at the following location:

```
$IASHOME/j2ee/home/applications/portal/portal/WEB_INF/
```

6.2.1.2 Parallel Page Engine Configuration Settings

The following table describes each of the different configuration parameters available for use with the Parallel Page Engine (PPE). Each parameter affects the

operation of the PPE in a different manner. Some are simply for logging, while others can affect the performance of the engine, or Oracle9*i*AS Portal itself. In most cases, the default values should be sufficient for use, however there may be configurations where this is not the case. Each parameter is described with it's syntax, description, and default

Table 6–1 Parallel Page Engine (PPE) parameters

PPE Setting	Syntax	Description	Default value

logmode	`<init-param>`
	`<param-name>logmode</param-name>`
	`<param-value>debug</param-value>`
	`</init-param>`
		Enables the Parallel Page Engine to run in debug mode. This mode will write debug information to the Parallel Page Engine log file. This mode does cause some degradation in performance, because large amounts of information are being written to disk. The Parallel Page Engine log file by default is located at:
		`$IASHOME/j2ee/home/application-deployments/portal/`
		Possible values are *debug*, *all*, and *perf* (for performance logging).
			`No Debugging`

poolSize	`<init-param>`
	`<param-name>poolSize</param-name>`
	`<param-value>25</param-value>`
	`</init-param>`
		This represents the number of connections that the Parallel Page Engine is capable of making at any one time. This value can be raised or lowered based upon performance needs. By setting the number higher, there will be more threads, and connections available for use, however this will use more resources.	`25`

Table 6–1 Parallel Page Engine (PPE) parameters

PPE Setting	Syntax	Description	Default value
requesttime	<init-param> <param-name>requesttime</param-name> <param-value>30</param-value> </init-param>	This is the default time out assigned to portlet requests which do not have there own time out value specified. It is applied as the amount of time (in seconds) allowed before response headers are returned by the server. Time outs are weighted by where they originate. If the portlet sets it's own time out value, then that is the time out which will be used. If no portlet time out is available, then the provider registration time out is used. If neither of these are present, then the requesttime is used.	30 sec
minTimeout	<init-param> <param-name>minTimeout</param-name> <param-value>5</param-value> </init-param>	This is the minimum timeout allowed to be used by a Portlet. Thus if the minTimeout is set to 5, and a portlet sends a timeout of 2, the minTimeout value of 5 would be applied to that portlet.	5 sec
stall	<init-param> <param-name>stall</param-name> <param-value>120</param-value> </init-param>	If the response headers are returned, but the data itself lags behind, then a stall comes into affect. This value keeps the Parallel Page Engine from holding onto connections forever. Once the response headers are received, the PPE makes every effort to wait as long as is feasible to retrieve all of the data. Set this value appropriately if the portlets being requested are large, or running over a slow network.	120 sec

Table 6–1 Parallel Page Engine (PPE) parameters

PPE Setting	Syntax	Description	Default value

httpsports	`<init-param>`
	`<param-name>httpsports</param-name>`
	`<param-value>433:444</param-value>`
	`</init-param>`
		This is a colon (':') separated list of ports on which the PPE should use SSL to communicate with the Portlet Repository.	`null`

prefix	`<init-param>`
	`<param-name>prefix</param-name>`
	`<param-value>/pls</param-value>`
	`</init-param>`
		The string used to indicate to Oracle HTTP Server where modplsql is located. The default matches the default Oracle9*i* Application Server installation configuration, but it must be changed if the Oracle9*i* Application Server configuration has changed.	`/pls`

proxyHost
proxyPort
	`<init-param>`
	`<param-name>proxyHost</param-name>`
	`<param-value>ph.comp.com</param-value>`
	`</init-param>`
	`<init-param>`
	`<param-name>proxyPort</param-name>`
	`<param-value>8888</param-value>`
	`</init-param>`
		This is the host name and port number of a proxy server that may be required to request data from the Oracle9*i* Application Server. These parameters are only required if a proxy server is in use between PPE and the iAS listener.	`N/a`

offlinePathHtml
offlinePathMxml
	`<init-param>`
	`<param-name>offlinePath</param-name>`
	`<param-value>/path/offline.html</param-value>`
	`</init-param>`
	`<init-param>`
	`<param-name>offlinePathMxml</param-name>`
	`<param-value>/path/offline.xml</param-value>`
	`</init-param>`
		By setting either of these, the PPE will be set to display the desired off-line message. There are two available messages: one for an HTML browser, and one for a mobile enabled device.	`null`

Table 6–1 Parallel Page Engine (PPE) parameters

PPE Setting	Syntax	Description	Default value

showError	`<init-param>`
	`<param-name>showError</param-name>`
	`<param-value>true</param-value>`
	`</init-param>`
		When a portlet times out, or something within the Parallel Page engine goes wrong with a particular portlet request, an error is displayed to the user. The messages tend to be generic, but do give the user some information and an indication that the page did not display as expected. By setting this to *false*, the user will not see these messages.	`true`

cacheBuffer	`<init-param>`
	`<param-name>cacheBuffer</param-name>`
	`<param-value>32768</param-value>`
	`</init-param>`
		The number of bytes to use for buffering when reading a completed page from the cache is set by using this parameter. By determining the size of pages generally used in a portal, this value can be adjusted to fit the portal configuration. By setting the value higher, a larger page can be read quickly, but more resources are needed. If the value is set low, then reading the cache file is slower.	`32768 Bytes`

cacheEncryption Key	`<init-param>`
	`<param-name>cacheEncryptionKey</param-name>`
	`<param-value>KEY</param-value>`
	`</init-param>`
		This key is used to obfuscate the headers used for caching using Oracle9*i*AS Web Cache. This allows for a more secure cache key, and makes retrieving a cached object more difficult for unwanted requests.	`Server Context information`

showPageDebug	`<init-param>`
	`<param-name>showPageDebug</param-name>`
	`<param-value>false</param-value>`
	`</init-param>`
		By setting showPageDebug to *true*, the Page timing information will be shown on every request.
		
		Refer to Section B.9, "Timing and Caching Statistics" for a description of the timing and caching statistics.
			`false`

Table 6–1 Parallel Page Engine (PPE) parameters

PPE Setting	Syntax	Description	Default value

dmsLogging	`<init-param>`
	`<param-name>dmsLogging</param-name>`
	`<param-value>false</param-value>`
	`</init-param>`
		By setting dmsLogging to *true*, the PPE will output data for DMS Logging.	`false`

queueTimeout	`<init-param>`
	`<param-name>queueTimeout</param-name>`
	`<param-value>10</param-value>`
	`</init-param>`
		The amount of time a request should stay in the queue before being timed out. This parameter can be used if requests for portlets are timing out, but the requests are never being sent. Although this points to other performance problems which could be solved by alternative configurations, this option is available to allow items to stay in the queue for longer or shorter periods of time.	`10 sec`

cacheDir	`<init-param>`
	`<param-name>cacheDir</param-name>`
	`<param-value>/PATH/</param-value>`
	`</init-param>`
		The cacheDir parameter value points to a directory in the file system where cache files are stored and retrieved. The PPE caches portlet contents and fully assembled page contents into this directory. There are no defaults for this parameter. Note that if it is not specified, the PPE will not use caching and the overall system performance will be greatly affected by it.	`none required`

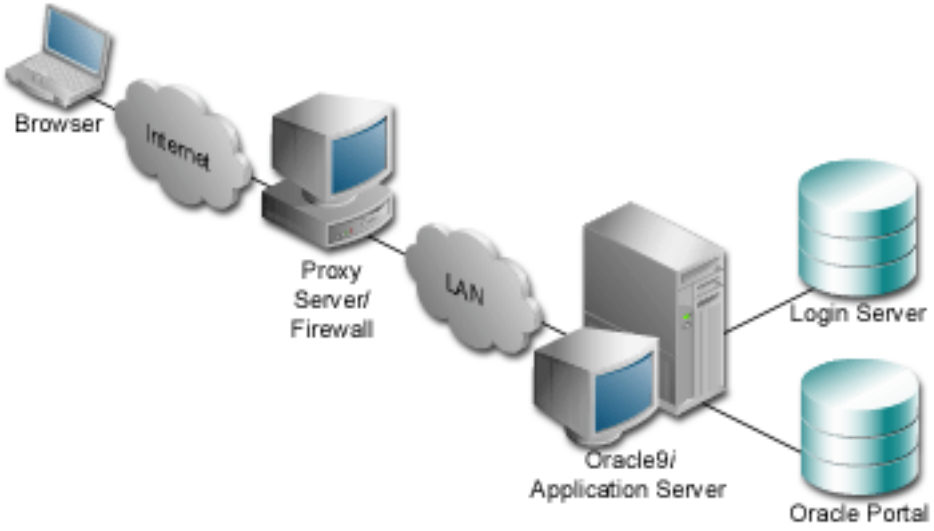
jspRoot	`<init-param>`
	`<param-name>jspRoot</param-name>`
	`<param-value>/JSP PATH/</param-value>`
	`</init-param>`
		The relative path where JSP files for JSP Pages can be found.	`jsp`

jspSrcAlias	`<init-param>`
	`<param-name>jspSrcAlias</param-name>`
	`<param-value></param-value>`
	`</init-param>`
		The Alias for the jsp engine, like /portal/jsp or some other path.	`/jsp/`

6.3 Configuring Reverse Proxy Servers Over the Internet

A reverse proxy server is a host process which is used as part of a firewall architecture to isolate the internal hosts from the externally accessible host(s) by providing a proxy through which external requests must pass to access internal services. Typically, such a proxy server takes the form of a dual-homed host. This means that it is a host with two network interface cards. One interface connects to the external network, and the other interface connects to the internal network, or demilitarized zone (DMZ) of the firewall.

Figure 6–1 Internet configuration with reverse proxy server



In this architecture, the browser accesses the server through a hostname which is published by the proxy server. The proxy server then forwards the request to the actual host within the firewall, which could be some other host name.

For this example, let's assume the following:

- The published address is `www.myportal.com`
- Internal to the firewall, the server name for the Oracle9*i* Application Server middle-tier is actually `server.company.com`.
- Externally, the server is addressed with the default port 80; however, internally, the `server.company.com` is listening on port 7777.

Complete these steps to configure Oracle9*i*AS Portal for this architecture:

1. Define configuration settings that allow the Oracle9*i* Application Server middle-tier to listen on port 7777, but assert to the server that it is using port 80.
2. Create `VirtualHost` entries that accept the internal host name, but then assert the externally visible host name, using the `ServerName` directive, so that self-referential URLs rendered on the Oracle9*i*AS Portal pages are valid for the browser.
3. Edit the hosts file on the internal middle-tier servers to define the IP addresses for the `ServerNames` asserted above, so that they can resolve the hostnames that are generated by Oracle9*i*AS Portal, for HTTP calls looping back to fetch portlet content.
4. Run the Oracle9*i*AS Portal Configuration Assistant (OPCA) in the MIDTIER mode to associate the reverse proxy server with Oracle9*i*AS Portal. Specify the following values:
   - `-host` *`reverse_proxy_hostname`*
   - `-port` *`reverse_proxy_port`*
   - `-chost` *`web_cache_host`*
   - `-cport_i` *`web_cache_invalidation_port`*

   Oracle9*i*AS Web Cache invalidation messages will now be sent to the Oracle9*i*AS Web Cache invalidation port. Note that the use of Oracle9*i*AS Web Cache is not depicted in the graphic that accompanies this example.

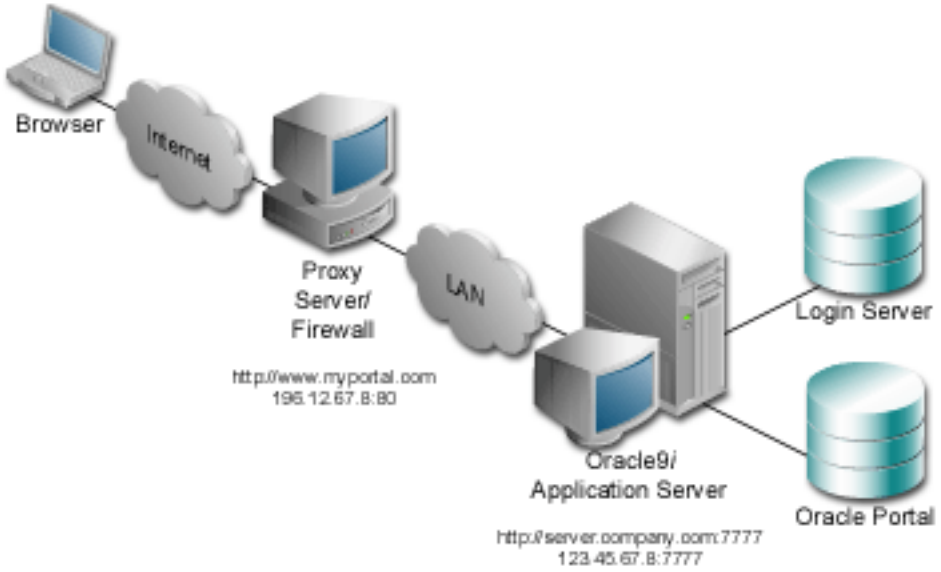
   See also: Section 3.2.2, "MIDTIER OPCA Mode" for more information on the OPCA MIDTIER mode.

5. Register the `www.myportal.com` domain name on a domain name server on the Internet, with IP address 196.12.67.8.

Note: The IP addresses used in this example are for illustration purposes only and may not be valid IP addresses.

The following figure shows the server names and ports used in the preceding example:

Figure 6–2 Example of reverse proxy server configuration



See also: For more information on how to set up proxy servers, refer to the white paper *A Primer on Proxy Servers*, available on:

`http://portalstudio.oracle.com`

6.3.1 Configuring the Oracle HTTP Server

You provide directives in the Oracle HTTP Server configuration file, `httpd.conf`, which specify the behavior described in the points above. The reverse proxy server contacts the internal middle-tier server as `server.company.com` over port 7777.

When the Oracle HTTP Server invokes the mod_plsql, and mod_jserv, it then passes `www.myportal.com` as the `ServerName` and port 80. URLs that are generated by Oracle9*i*AS Portal code use `www.myportal.com` and port 80.

The directive "useCanonicalName on" instructs Oracle HTTP Server to use the `ServerName` specified. Otherwise, it passes the name used in the Host field of the request.

The relevant sections in the `httpd.conf` file for the `server.company.com` Oracle9*i* Application Server configuration are shown in the following.

```
### Section 2: 'Main' server configuration
```

```
...
Port 80
Listen 7777
Listen 80

ServerName www.myportal.com
...
UseCanonicalName On
...
### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on
# your machine you can setup VirtualHost containers for them.
#
# If you want to use name-based virtual hosts you need to define at
# least one IP address (and port number) for them.
#

# This section is mandatory for URLs that are generated by
# the PL/SQL packages of the Oracle Portal
# These entries dictate that the server should listen on port
# 7777, but will assert that it is using port 80, so that
# self-referential URLs generated specify www.myportal.com:80
# This will create URLs that are valid for the browser since
# the browser does not directly see the host server.company.com.
NameVirtualHost 123.45.67.8:7777

<VirtualHost server.company.com:7777>
ServerName www.myportal.com
Port 80
</VirtualHost>

# Since the previous virtual host entry will cause all links
# generated by the Oracle Portal to use port 80, the server.company.com
# server needs to listen on 80 as well since the Parallel Page
# Engine will make connection requests to Port 80 to request the
# portlets.

NameVirtualHost 123.45.67.8:80

<VirtualHost server.company.com:80>
ServerName www.myportal.com
Port 80
</VirtualHost>
```

If you need to support multiple aliases for the published address `www.myportal.com`, then some additional directives are needed. For example, if the server also needs to be accessible as `www.portal.com`, then you need to define additional virtual host entries on the internal server. This is so the reverse proxy directs requests from each corresponding published hostname to a related internal host alias which can assert the correct published name.

In this example, the VirtualHost sections appear as follows:

```
NameVirtualHost 123.45.67.8:7777

<VirtualHost server.company.com:7777>
ServerName www.myportal.com
Port 80
</VirtualHost>

<VirtualHost server2.company.com:7777>
ServerName www.portal.com
Port 80
</VirtualHost>

NameVirtualHost 123.45.67.8:80

<VirtualHost server.company.com:80>
ServerName www.myportal.com
Port 80
</VirtualHost>

<VirtualHost server2.company.com:80>
ServerName www.portal.com
Port 80
</VirtualHost>
```

6.3.2 Resolving Domain Names

A local HOSTS file can help resolve domain names that are not normally visible to the internal network. For example, the Oracle9*i* Application Server host for `server.company.com` makes requests to itself, but the URLs that it is requesting are referring to `www.myportal.com`. You must create host entries in the local HOSTS file on that machine allowing it to resolve this name, within the firewall. The hosts entry for this example should include the following lines:

```
# This is a sample HOSTS file used by Microsoft TCP/IP
# for Windows NT/2000.
#
```

```
127.0.0.1    localhost
123.45.67.8  www.myportal.com
```

If you do not provide these entries in the local HOSTS file, then you need to set the Oracle9*i* Application Server host to recognize a proxy server that would take the request out to the Internet and back in through the reverse proxy (`www.myportal.com`). Avoid this setup as this may result in poor performance.

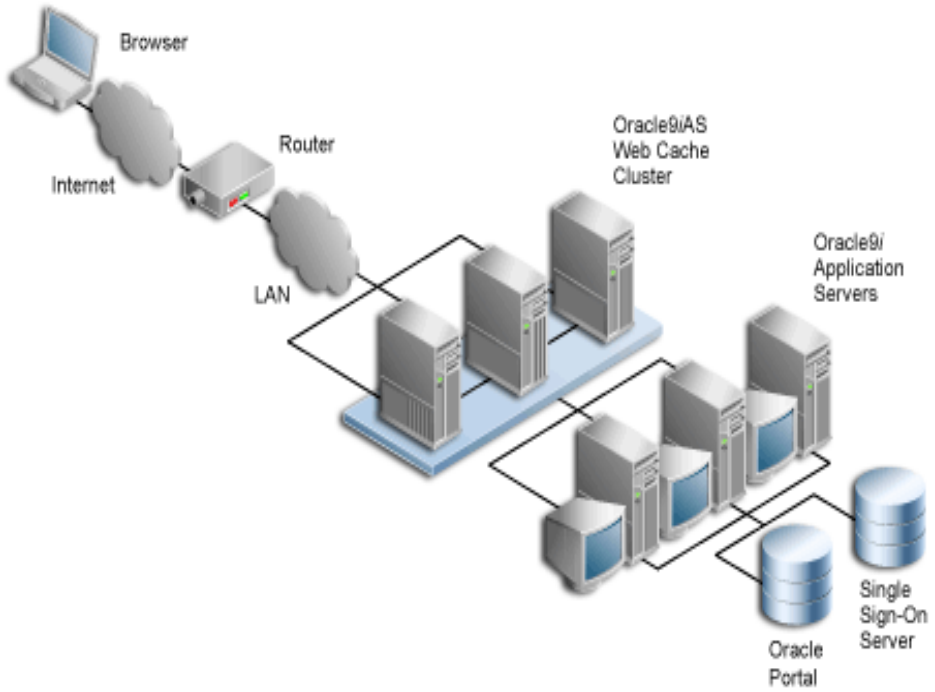
Note: On some platforms, such as HP, there is a file which indicates the search order that should be applied to the sources for IP name mapping. For the preceding example to work, if such a file exists on your platform, make sure that it specifies the local hosts file to be checked for IP mapping, before any DNS servers.

6.4 Configuring Load Balancing Routers

The purpose of a Load Balancing Router (LBR) is to provide a single published address to the client browsers, and provide a "farm" of Web servers which actually service the requests, based on the distribution of the requests done by the LBR. The LBR itself is a very fast network device which can distribute Web requests to a large number of physical servers.

If you want to install multiple Oracle9*i* Application Server middle-tier servers to handle a large load, you could configure Oracle9*i*AS Portal as illustrated in the following diagram:

Figure 6–3 Load balancing router configuration



This example shows that the Load Balancing Router (LBR) balances the load between all three instances of the Oracle9*i*AS Web Cache cluster. each one of the Oracle9*i*AS Web Cache clusters can in turn load balance any of the middle-tier servers which communicate with Oracle9*i*AS Single Sign-On Server and Oracle9*i*AS Portal. In this example, let's call the three Oracle9*i*AS Web Cache instances wc1, wc2, and wc3, and let's call the Oracle9*i* Application Server middle-tier servers svr1, svr2, and svr3. So in the above example wc1 can load balance svr1, svr2, as well a svr3. Additionally wc2 and wc3 can do the same.

All the Oracle9*i* Application Server middle-tier servers must have DAD entries for each of the databases. A good way to accomplish this is to have the middle-tier servers share a file system that contains the configuration information for the DADs, and allows them to share cache files.

The important points to consider with this configuration include:

- The Internet DNS maps the name `www.myportal.com` to the external IP address on the LBR.

- The LBR performs load balancing of requests to `www.myportal.com` to s`vr1.company.com`, `svr2.company.com`, and `svr3.company.com`, addressing the request to their IP addresses, but still containing `www.myportal.com` in the Host: field of the HTTP request.
- Each of the middle-tier hosts accepts requests to `www.myportal.com`, and their `httpd.conf` files assert that name as the ServerName. Hence the names svr1, svr2, and so on are irrelevant.
- Unless your LBR does port mapping, you should configure the internal servers to use the same ports as the LBR.
- Optimal cache utilization can be realized by mounting a shared file system on which to write the cache files. If you decide not to have the middle-tier servers share a cache directory, caching will still work, but with a lower hit ratio

6.4.1 Additional Configuration Related to Oracle9*i*AS Web Cache Invalidation

Some additional configuration is needed, because invalidation requests need to be returned properly to Oracle9*i*AS Web Cache and not to the LBR.

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library.

If the LBR that is in use can listen on multiple ports and if it also has the ability to route the invalidation messages back to Oracle9*i*AS Web Cache, then OPCA needs to be run in the MIDTIER mode, specifying the following settings:

- `-host` LBR host name
- `-port` LBR port name
- `-chost` LBR host name
- `-cport_i` LBR port name

If the LBR is not able to listen on multiple ports, or if it does not have the ability to route the invalidation messages back to Oracle9*i*AS Web Cache, another LBR, which we will call the invalidation request handling LBR, needs to be set up to handle the routing of the invalidation messages. In this case, OPCA needs to be run in the MIDTIER mode, specifying the following settings:

- `-host` LBR host name
- `-port` LBR port name
- `-chost` invalidation request handling LBR host name

- `-cport_i` invalidation request handling LBR port name

Instead of using another LBR as an invalidation request handling LBR, you could also use one of the Oracle9*i*AS Web Cache instances, in the Oracle9*i*AS Web Cache cluster to handle the invalidation (for example wc1). This instance must be part of the Oracle9*i*AS Web Cache cluster, but it is recommended that you do not cache anything, or handle HTTP requests with this designated Oracle9*i*AS Web Cache instance, in order to increase availability. In this case, OPCA needs to be run in the MIDTIER mode, specifying the following specific settings:

- `-host` LBR host name
- `-port` LBR port name
- `-chost` hostname of the designated Oracle9*i*AS Web Cache cluster instance.
- `-cport_i` invalidation port of the designated Oracle9*i*AS Web Cache cluster instance

Once one of the above options is configured correctly, the invalidation messages will be routed correctly back to Oracle9*i*AS Web Cache and the benefits from using the LBR, as well as Oracle9*i*AS Web Cache can be enjoyed.

6.5 Setting up Two Sites Using the Same Infrastructure

If you want to configure two sites to use the same Portal repository, some additional Oracle9*i*AS Web Cache configuration is required.

Assuming that you would want to add another site called `www.myportal2.com` to the configuration shown in Figure 6–3, "Load balancing router configuration". `www.myportal2.com` needs to be mapped to the same LBR.

Assuming that you have already defined `www.myportal.com` as a site in Oracle9*i*AS Web Cache, the additional configuration consists of creating a site alias in Oracle9*i*AS Web Cache.

It is important that `www.myportal.com` is set up as a site, while `www.myportal2.com` has to be defined as a site alias. This way, the invalidation messages sent to Oracle9*i*AS Web Cache invalidate content that is cached against both sites. This configuration makes the multiple site definitions transparent to the Portal repository.

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* for information on how to configure a site and a site alias.

6.6 Configuring Multiple Middle-Tiers to Use the Same Infrastructure

Multiple Oracle9*i*AS Portal middle-tier installations associated to a single Oracle9*i* Application Server infrastructure do not work properly in a default Oracle9*i* Application Server installation. There are a number of integration points that need to be addressed in case you try to use such a configuration:

Oracle9*i*AS Portal and Oracle9*i*AS Web Cache

Detailed steps for setting up Oracle9*i*AS Web Cache in a multiple middle-tier environment are described in Section 6.8.8, "Associating Multiple Middle-Tiers to a Single Infrastructure".

Oracle9*i*AS Portal and OID

Every Oracle9*i*AS Portal middle-tier installation drops and recreates the Portal users in OID. This means that the Oracle9*i* Application Server instance password of the last run middle-tier installation should be used for Portal runtime access.

After all the middle-tier installations are performed, the users can change the Portal user password in OID. This does not require any other changes in the Portal repository.

Oracle9*i*AS Portal and Oracle9*i*AS Wireless

If Oracle9*i*AS Wireless is configured with Oracle9*i*AS Portal during the middle-tier installation, the middle-tier install registers the Portal on the Oracle9*i*AS Wireless service. In case of multiple midtier installs, the last configured Oracle9*i*AS Wireless service URL is stored in the Oracle9*i*AS Portal instance. You can change this to your choice of Oracle9*i*AS Wireless service by running the following scripts in the Oracle9*i* Application Server middle-tier selected for the Oracle9iAS Wireless service:

UNIX:

```
ORACLE_HOME/wireless/sample/portalRegistrar.sh
```

Windows:

```
ORACLE_HOME/wireless/sample/portalRegistrar.bat
```

Portal Provider UI Framework

Multiple Portal middle-tier installations will overwrite the existing Default JPDK Instance URL which is used for creating the Providers. The users can change this to their choice of JPDK Instance URL using the following steps :

1. Login to Portal in the browser.
2. Click on the Builder link.
3. Click on the Administrator tab. Click on Global Settings in the Services portlet.
4. Click on the Configuration tab. Enter the Default JPDK Instance URL of any installed Portal middle-tier.

6.7 Tuning the Oracle HTTP Server

However you choose to configure the Oracle HTTP Server listener, you can optimize performance by setting an approximate number of simultaneous requests that can be handled by the Oracle HTTP Server listener.

On UNIX, in particular, since the Oracle HTTP Server is process-based, each process needs to open a database connection for each DAD that has requested it. As a result, the number of requests can be quite high, which may result in clients being "locked out" if the number of sessions allowable has been exceeded. However, setting too high of a value unnecessarily consumes resources.

The scenario is described below:

1. For every service request from an Oracle9*i*AS Portal DAD, there is one network connection and two sessions (the two sessions use the same physical connection).

   The first session is for "portal" and the second is for "portal_public".

2. If you are logging into Oracle9*i*AS Portal, then you'll need to open a connection for the Oracle9*i*AS Single Sign-On Server DAD (SSO DAD). This will consume one network connection and two sessions.

   In this case, the first session is for "orasso" and the second session is for "ora_sso_public".

3. The Oracle HTTP Server configuration setting that determines the maximum number of requests being handled simultaneously is named `MaxClients`. It defaults to 150.

If each user were logging in and working in Oracle9*i*AS Portal, then scenario (1) and (2) above would result in four sessions per process. The total number of sessions for such a scenario is calculated as follows:

```
150 * 4 = 600
```

600 sessions and approximately 300 database connections (2 sessions per connection)

6.7.1 Configuring the MaxClient Setting

Since login frequency is generally lower than Oracle9*i*AS Portal access frequency, it makes sense to configure the Oracle9*i*AS Single Sign-On Server on a separate Oracle HTTP Server listener. The objective is to tune down the `MaxClient` setting to a value that is reasonable without affecting the needs of the portal system.

Oracle9*i*AS Portal makes extensive use of mod_plsql which maintains a pool of connections to the database. The `MaxClient` parameter tunes the number of processes which directly relates to the number of database connections pooled by mod_plsql.

> **See also:** *Oracle9i Application Server mod_plsql User's Guide*, which is provided as part of the Oracle9*i* Application Server documentation library.

Follow these steps to configure the MaxClient setting:

1. For the Oracle9*i*AS Single Sign-On Server's listener, once you've determined the approximate value to set for the `MaxClients` parameter, edit this accordingly in the configuration file, `httpd.conf`, which is located in:

   ```
   ORACLE_HOME/Apache/Apache/conf/
   ```

   Tune down the `MaxClients` setting to control the number of requests that Oracle HTTP Server services on the Oracle HTTP Server listener. This ultimately controls the maximum number of sessions that could be established.

2. For the Oracle9*i*AS Portal listener, you can separately tune the `MaxClients` parameter according to the needs of the Oracle9*i*AS Single Sign-On Server and the needs of Oracle9*i*AS Portal, without affecting each other. This parameter directly corresponds to the number of sessions established and to the maximum workload that the Oracle HTTP Server listener can handle on the Portal listener.

   The MaxClients section in the `httpd.conf` file is shown below:

```
# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# the system with it as it spirals down...
#
MaxClients 150
```

Note:

- If you tune separately, you will have a separate listener for Oracle9*i*AS Portal and the Oracle9*i*AS Single Sign-On Server. The former controlling the resources (sessions) on the portal database and the latter controlling the resources on the Oracle9*i*AS Single Sign-On Server database.
- The number of sessions and connections that the database permits is limited by the value set in the Oracle9*i* database's `init.ora`. Refer to the Oracle9*i* database documentation library for more information.

6.8 Configuring Oracle9*i*AS Portal to Work with Oracle9*i*AS Web Cache

This section will discuss how to configure Oracle9*i*AS Portal to work with Oracle9*i*AS Web Cache. This section includes the following subsections:

- Evaluating the Oracle9iAS Web Cache Logs
- Oracle9iAS Web Cache Configuration Scripts
- Troubleshooting the Oracle9iAS Web Cache Configuration
- Accessing the Oracle9iAS Web Cache Administration Portlet
- Tuning Oracle9iAS Web Cache
- Disabling Oracle9iAS Web Cache
- Shutting down Oracle9iAS Web Cache completely
- Associating Multiple Middle-Tiers to a Single Infrastructure

See also: *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library.

6.8.1 Evaluating the Oracle9*i*AS Web Cache Logs

Log files for Oracle9*i*AS Web Cache are typically stored in `ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows.

There are two log files:

- The access_log file.
- The event_log file.

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library.

6.8.2 Oracle9*i*AS Web Cache Configuration Scripts

You can configure Oracle9*i*AS Portal to work with Oracle9*i*AS Web Cache in a variety of ways, but some scripts are supplied to facilitate this. These scripts are covered in more detail in Section B.2, "Oracle9iAS Web Cache configuration scripts".

The scripts covered in Appendix B are:

- `cachseed.sql`, which allows you to modify all the Oracle9*i*AS Web Cache configuration parameters.
- `cachset.sql`, which allows you to turn the use of Oracle9*i*AS Web Cache on or off. Information on how to disable Oracle9*i*AS Web Cache completely is covered as well.
- `cachjsub.sql`, which allows you to manage the invalidation message processing job.

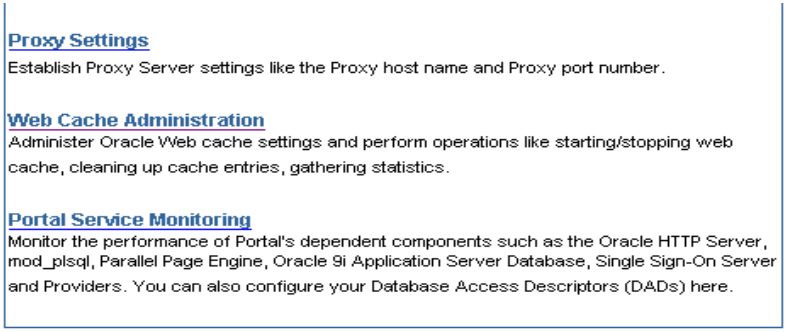
6.8.3 Troubleshooting the Oracle9*i*AS Web Cache Configuration

> **See also:** Troubleshooting information is covered in Section 8.1, "Oracle9iAS Web Cache related issues".

6.8.4 Accessing the Oracle9*i*AS Web Cache Administration Portlet

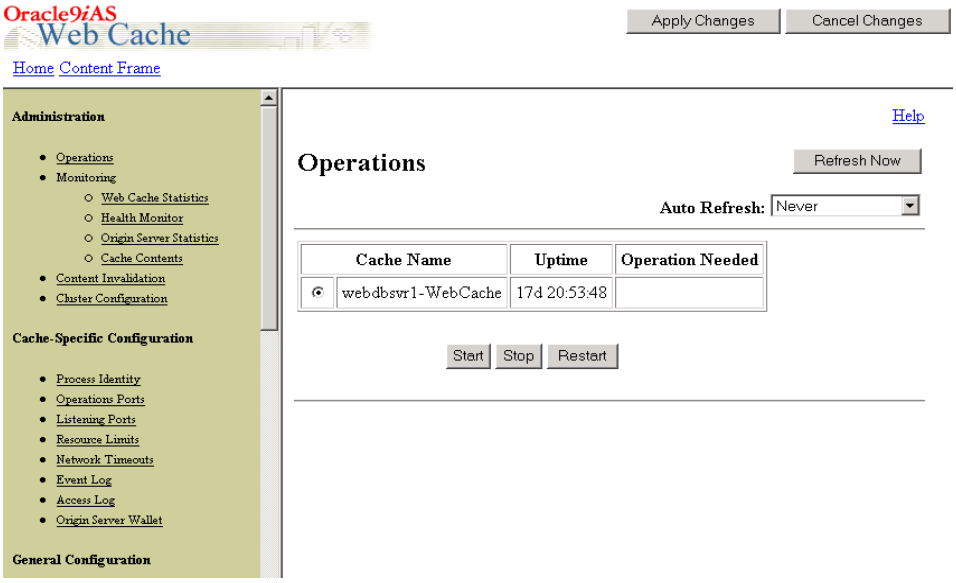
You can access the Oracle9*i*AS Web Cache administration portlet directly from Oracle9*i*AS Portal. In the **Services** portlet, go to the Proxy Settings page. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's **Administer** tab as shown in the following image.

Figure 6–4 The Oracle9iAS Web Cache administration portlet



After entering the Oracle9*i*AS Web Cache administrator username and password, you will be able to use the web-based Oracle9*i*AS Web Cache administration tool, as shown in the image below.

Figure 6–5 The Oracle9iAS Web Cache administration tool



6.8.5 Tuning Oracle9*i*AS Web Cache

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library.

6.8.6 Disabling Oracle9*i*AS Web Cache

Follow these steps if you want to disable the use of Oracle9*i*AS Web Cache:

1. In the **Services** portlet, click on Global Settings. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's **Administer** tab.
2. Click on the Cache tab.
3. Uncheck the "*Enable Web Cache for caching portal content*" check box.

> **Note:** By doing this, you effectively stop using Oracle9iAS Web Cache in Oracle9iAS Portal. However, requests to and from the middle-tier will still pass through Oracle9iAS Web Cache.
>
> If you are concerned about the performance loss that is caused by this extra step, you can also shut down Oracle9iAS Web Cache completely as shown in Section 6.8.7, "Shutting down Oracle9iAS Web Cache completely".

6.8.7 Shutting down Oracle9*i*AS Web Cache completely

If you want to shut down Oracle9*i*AS Web Cache completely, you must perform the following steps:

1. Run *cachset.sql* as mentioned in Section B.2.2, "Using cachset.sql".
2. Shut down the Oracle9*i*AS Web Cache instance.

> **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library for information about shutting down Oracle9*i*AS Web Cache.

1. Open the `httpd.conf` file and change the Listen directive to the port on which Oracle9*i*AS Web Cache was listening. this file is located in:

   ```
   ORACLE_HOME/Apache/Apache/conf/
   ```

2. Restart the Oracle HTTP Server.

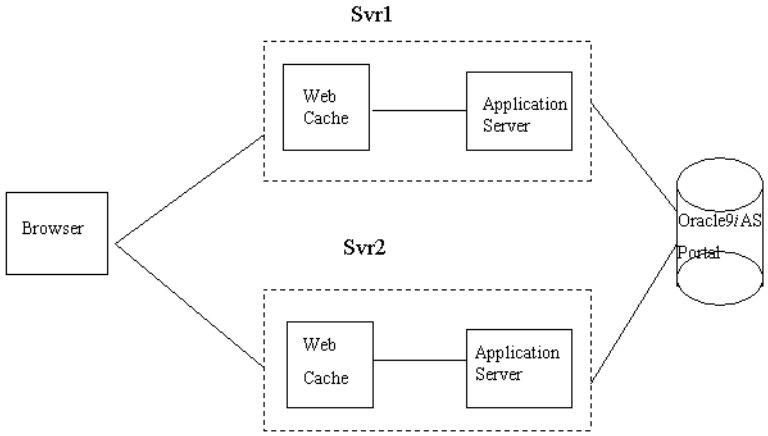
See also: *Oracle9i Application Server Administrator's Guide* in the Oracle9*i* Application Server documentation library for information about shutting down the Oracle HTTP Server.

6.8.8 Associating Multiple Middle-Tiers to a Single Infrastructure

Multiple Oracle9*i*AS Portal middle-tier installations associated to a single Oracle9*i* Application Server infrastructure do not work properly in a default Oracle9*i* Application Server installation. The reason for this is that Oracle9*i*AS Portal sends invalidation messages to a single Oracle9*i*AS Web Cache instance. Entries for the location of the Oracle9*i*AS Web Cache host, the invalidation and administrator port are stored within the Oracle9*i*AS Portal instance. Every Portal middle-tier installation (performed with the Oracle9*i*AS Portal Configuration Assistant (OPCA) in MIDTIER mode) overwrites these entries. This means that only the last installed portal middle-tier will work properly, because invalidation messages for the previous installation are now sent to the wrong Oracle9*i*AS Web Cache. This can result in stale content, as the content in the previously installed middle-tier will never get invalidated properly.

The following image shows this scenario:

Figure 6–6 Multiple Middle-Tiers Associated to a Single Infrastructure



When the OPCA is run in MIDTIER mode on Svr2, everything is set up to work fine. If after that, however, the OPCA is run in MIDTIER mode for Svr1, the settings that were previously stored for Svr2 in Oracle9*i*AS Portal are overwritten.

There are two ways to address this issue:

1. Disable the use of Oracle9*i*AS Web Cache in Oracle9*i*AS Portal.

   You can disable Oracle9*i*AS Web Cache from within Oracle9*i*AS Portal by following the steps explained in Section 6.8.6, "Disabling Oracle9iAS Web Cache".

   Alternatively, if you are unable to login through Svr2 into the Portal due to this misconfiguration, you can run *cachset.sql* script to disable the use of Oracle9*i*AS Web Cache in the Portal, as described in Section B.2.2, "Using cachset.sql".

   Note: Do not forget to clear the File System Cache after disabling the use of Oracle9*i*AS Web Cache, in order to make sure that the served content is up-to-date.

   ---

2. Shut down Oracle9*i*AS Web Cache on one of the middle-tiers.

   Choosing this option will allow you to still use Oracle9*i*AS Web Cache on one of the middle-tiers, taking at least partial advantage of the benefits that Oracle9*i*AS Web Cache has to offer.

   In the example shown, you could, for example, shut down Oracle9*i*AS Web Cache on Svr2. This is better than disabling the use of Oracle9*i*AS Web Cache, because it allows invalidation messages to still be sent to Svr1. Follow the instructions in Section 6.8.7, "Shutting down Oracle9iAS Web Cache completely" to shut down one of the Oracle9*i*AS Web Cache instances.

7

Configuring the Search Features in Oracle9*i*AS Portal

This chapter provides information on setting up the built-in Oracle Text search capabilities in Oracle9*i*AS Portal page groups.

This chapter contains the following sections:

See also:

For Oracle Text documentation, see your Oracle9*i* database documentation library, and for Oracle Text technical papers, training materials, code samples, and so on, visit:

http://otn.oracle.com/products/text/

7.1 New Search Features

This release of Oracle9*i*AS Portal includes the following new features for search:

- Search results are displayed in a portlet. The portlet can be added to any portal page (a default Search Results page is provided). The portlet can be customized to control the information that appears in each listing.
- Extensive customization possibilities for the search submission portlet. This allows for customizing the way that search result items are rendered. You can customize the way that the submission form is rendered as well as the way that the results are rendered.
- The ability to show the search results in a search portlet on another page, such as the pre-defined search results page, or in the same portlet as the search submission.
- Autoquery. This feature allows you define a set of search conditions and have that search executed automatically when a portlet is rendered.
- Oracle Ultra Search Integration. Oracle Ultra Search lets you index and search Oracle9*i*AS Portal portals, as well as web sites, database tables, files, mailing lists, and user-defined data sources. As such, you can use Oracle Ultra Search to build different kinds of search applications. An Oracle Ultra Search portlet is provided with this release of Oracle9*i*AS Portal as well.

To access these new features, click **Search Setting**s in the Services portlet. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's Administer tab. To access the new customization features, it is necessary to use the *Edit Defaults* customization settings on an instance of a search portlet.

7.2 Prerequisites

You must be logged on as a portal administrator to configure Oracle Text (Formerly Intermedia Text), and create, alter, update, and drop Oracle Text indexes.

Before using Oracle Text in Oracle9*i*AS Portal, perform the following tasks:

- Install and configure the Oracle9*i* database (minimum 9.0.1.3) with the Oracle Text option by running the Oracle Universal Installer (OUI) and the Database Configuration Assistant.

  See also: Oracle9*i* installation documentation for more information about installing and configuring an Oracle9*i* database to use Oracle Text.

7.3 Searching in Oracle9*i*AS Portal

The main points to know when searching in Oracle9*i*AS Portal page groups include:

- Searches on portal pages are performed from the Oracle9*i*AS Portal Navigator and can item, page, category or perspective search results.
- Search is by default enabled for *all* page groups created in your Oracle9*i*AS Portal installation. It cannot be enabled on one page group and disabled on another page group. Search results are not restricted to the page group on which the search is performed. A search can be performed across all page groups or within a single page group. The default for a basic search portlet is to search all page groups.

There are three levels and two modes of searching in Oracle9*i*AS Portal page groups depending on the type of search you use: basic search, advanced search, custom search and whether Oracle Text search is enabled. Only allows you to enter the main search term.

7.3.1 Basic Search

The basic search is available from the Search field on the navigation bar. It only allows you to enter the main search term.

This type of search looks for the specified words in the item attributes such as the display name, description, and keywords of items, as well as the display name and description of folders, categories, and perspectives. Depending on whether *Oracle Text* is enabled, the basic search can also look in the content of documents and URLs. A search results page displays all items that meet the search criteria.

> **See also:** The Oracle9*i*AS Portal Online Help topics: *Performing a basic search*, *Setting up the search feature*, and *Editing navigation page properties*.

When Oracle Text is not enabled in the basic search:

- Searches are performed only on the item attributes *title*, or display name, *description*, *keywords* and *author*.
- Searches are performed on the page attributes, such as *title*, *description* and *keywords*.
- Searches are performed on the *perspective* and *categories* attributes, such as *title* and *description*.
- The search automatically uses wildcards.

Example:

If you search on "sing,", the results of the search include: sing, single and tossing.

When Oracle Text is enabled in the basic search:

- Searches are performed on the item attributes *author, creator, description*, *itemtype, keywords, name, title* and *updator.*
- Searches on the following page attributes: *creator, description*, *keywords*, *name*, *title* and *updator.*
- Searches are performed on the following perspective attributes: *name, title* and *description*.
- Searches are performed on the following categories attributes: *name, title* and *description*.
- The search uses STEM searching.

  Examples:

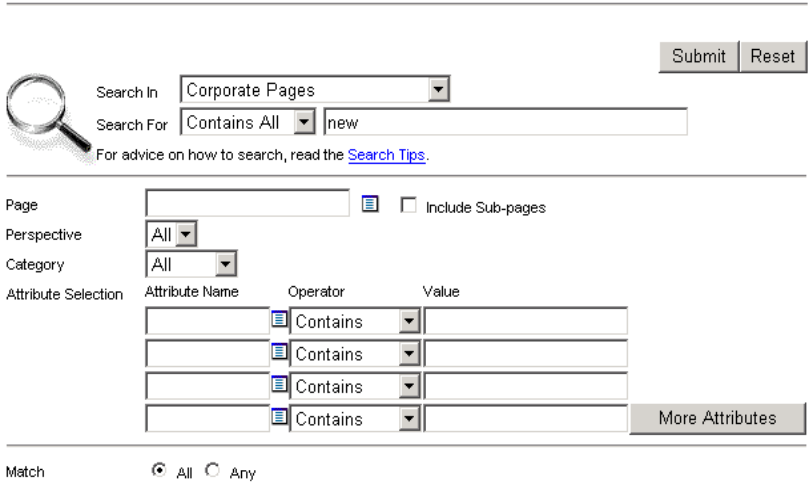
  If you search on "distinguish," the results of the search include: distinguish, distinguished, distinguishes. If you search on "sing," the results of the search include: sing, sang, sung.

7.3.2 Advanced Search

With advanced search, which is always enabled, you can:

- Find content that contains any of the specified words.
- Search across a page group other than the current page group, or across all page groups.
- Restrict the search to a particular folder, category, perspective, item type, or attribute.
- If Oracle Text is enabled, for near, soundex, and fuzzy items are also searched.

Figure 7–1 Advanced Search in Oracle9iAS Portal



See also: The Oracle9*i*AS Portal Online Help topic: *Performing an Advanced Search.*

When Oracle Text is not enabled in the advanced search:

- Searches are only performed on the item attributes *title*, or display name, *description*, *keywords* and *author.*
- Searches are performed on the page attributes, such as *title*, *description* and *keywords.*
- Searches are performed on the perspective and categories attributes *title* and *description*.
- The search automatically uses wildcards.

When Oracle Text is enabled in the advanced search:

- Searches are performed on the item attributes *author, creator, description*, *itemtype, keywords, name, title* and *updator.*
- Searches are performed on the page attributes *creator, description*, *keywords*, *name*, *title* and *updator.*
- Searches are performed on the following perspective attributes: *name, title* and *description*.

- Searches are performed on the following categories attributes: *name*, *title* and *description*.
- The search uses STEM searching.

7.3.3 Custom Search

The custom search is available from a Custom Search portlet. The custom search portlet is fully customizable and can be modified to search on any of the item, page, categories and perspective attributes.

> **See also:** For more information on custom searching, refer to the Oracle9*i*AS Portal Online Help topic *Performing a custom search.*

When Oracle Text is not enabled in the custom search:

- Searches are only performed on the item attributes *title*, or display name, *description*, *keywords* and *author.*
- Searches are performed on the page attributes, such as *title*, *description* and *keywords*.
- Searches are performed on the perspective and categories attributes *title* and *description*.
- The search automatically uses wildcards.

When Oracle Text is enabled in the custom search:

- Searches are performed on the item attributes *author*, *creator*, *description*, *itemtype*, *keywords*, *name*, *title* and *updator.*
- Searches are performed on the page attributes *creator*, *description*, *keywords*, *name*, *title* and *updator.*
- Searches are performed on the following perspective attributes: *name*, *title* and *description*.
- Searches are performed on the following categories attributes: *name*, *title* and *description*.
- The search uses STEM searching.

7.3.4 STEM Searching

By default STEM searching is used when Oracle Text is enabled. However, STEM searching is only used when logged in to Oracle9*i*AS Portal in one of the languages

where STEMing is supported in Oracle Text. STEMing is supported in the following languages:

```
AMERICAN
CANADIAN FRENCH
DUTCH
ENGLISH
FRENCH
GERMAN DIN
GERMAN
ITALIAN
LATIN AMERICAN SPANISH
MEXICAN SPANISH
SPANISH
```

In all other languages, the STEM operator is not used.

7.3.5 Oracle Text

As discussed earlier, Oracle9*i*AS Portal has built-in support for Oracle Text indexing. It is worth repeating that search is enabled for *all* page groups created in your Oracle9*i*AS Portal installation. It cannot be enabled on one page group and disabled on another page group. The search is performed on the actual content in documents such as PDF, PowerPoint, and Word as well as the contents on URL pages, text, and HTML.

If Oracle Text is *not* enabled, end users can always perform a basic or advanced search in the page group.

Note: For Items and Pages, where custom attribution is supported, the main search term is matched against the following when Oracle Text is enabled:

- A *text* datatype
- custom attributes
- The contents of any files or URLs, related to the item or page as file or URL item.

7.3.6 Viewing Oracle Text Search Results

If themes and gists are enabled from the Search Settings page (see Figure 7–3, "Services portlet Oracle Text properties"), then you can access the themes and gists for documents returned by a search from the search results. You can:

- View an HTML version of the file.
- View an HTML version of the file with search terms highlighted in any color or font set by the Oracle9*i*AS Portal administrator.
- View major themes in a chart (document theme analysis).
- View a short summary about the contents of the file (gist).

7.4 Oracle Text Performance

Oracle Text performance may be affected by the following query, indexing, and update considerations:

7.4.1 Query Considerations

How does the size of my data affect queries?

The speed at which the text index can deliver ROWIDs is not affected by the actual size of the data, but by the size of the Token Table which holds the list of words, and information about the rows in which they appear. Text query speed will be related to the number of rows that must be fetched from this Token Table, and the length of each row.

Thus, it should be nearly as fast to find a rare word in a large document set as it is to find a common word (or many uncommon words) in a smaller document set.

How does the source type of my data affect queries?

The format of the documents (for example, plain ASCII text, HTML or Microsoft Word) should make no difference to query speed. The documents are filtered to plain text at indexing time, not query time.

The "cleanliness" of the data makes a difference. Spell-checked and sub-edited text for publication tends to have a much smaller total vocabulary (and therefore size of token table) than informal text such as e-mails, which contain many spelling errors and abbreviations to bloat the token table.

7.4.2 Indexing Considerations

How long should indexing take?

Indexing text is a resource-intensive process. Obviously, the speed of indexing depends on the power of the hardware involved, but you should expect somewhere between 50MB per hour on workstation-class Windows NT/2000 machine (approximately 400MHz CPU, 128MB memory) to more than 1GB per hour on a large multi-CPU, multi-gigabyte server machine. The latter figure assumes you are using parallel indexing on a partitioned table.

For most real-life systems, the time to index a complete table of documents will be measured in hours, and in some cases days.

How do I track the progress of the indexing process?

You can use the `ctx_output.start_log` *filename* command to log output from the indexing process. The *filename* will normally be written to `ORACLE_HOME/ctx`, but you can change the directory using the `log_directory` parameter in `ctx_adm.set_parameter`.

Otherwise, for a course-grained answer, you can count the number of rows in the `DR$xxx$K` table. There will be one row in here for each row that has been indexed. However, these rows are only committed when the indexing process runs out of indexing memory and does a "flush" to the database. It is even possible that this will never happen until indexing is complete.

How much disk space overhead will the indexing require?

The overhead (the amount of space needed for the `DR$` index tables) varies between approximately 25% of the original text volume, and 100%. Generally, the larger the total amount of text, the smaller the overhead, but many small records will use more overhead than fewer large records. Also, "clean" data (such as published text) will require less overhead than "dirty" data such as e-mails or discussion notes, since the "dirty" data is likely to include many unique words from misspellings, abbreviations, and so on.

Theme indexes are generally much smaller than text indexes. Creating a theme index only will generally require very little storage, but creating a text index only will not save you much space over a combined index, though it is likely to be significantly faster.

How does the format of my data affect indexing?

Looking at indexing overhead, you can expect much lower overheads for formatted documents (for Microsoft Word files) since such documents tend to be very large compared to the actual text held in them.

So 1GB of Word documents might only require 50MB of index space, whereas 1GB of plain text might require 500MB, since there is ten times as much "plain text" in the latter set.

Indexing time is harder to determine. Although the reduction in the amount of text to be indexed will have an obvious effect, we must balance this out against the cost of filtering the documents. In general, these will roughly cancel out, so the time to index 1GB of formatted docs will be about the same as to index 1GB of plain text, although it may be a little longer.

7.4.3 Update Considerations

How often should I index new or updated records?

How often do you need to? The less often you run re-indexing then the less fragmented your indexes will be, and the less you will need to optimize them. However, this means that your data will become progressively more out of date, which may be unacceptable for your users.

Many systems can handle overnight indexing. This means data that is less than a day old is not searchable. Other systems use hourly, ten minute, or five minute updates.

To keep your indexes up to date, so you can search on recently added content, you will need to use the procedure wwv_context.sync(), which in turn calls the ctx_ddl.sync_index procedure to synchronize the six Oracle9*i*AS Portal indexes.

See also:

- Section 7.5.4.1, "Synchronizing Oracle9iAS Portal Text Indexes"
- The Oracle Text documentation in the Oracle9*i* database documentation library.

How can I tell when my indexes are getting fragmented?

Synchronizing indexes can cause them to become fragmented. Heavily fragmented text indexes can cause the search query performance to deteriorate. To rectify this it is necessary to optimize the indexes. This is done by using the call procedure

```
wwv_context.optimize();
```

Which in turn calls the *ctx_ddl.optimize_index* procedure.

The *wwv_context.optimize* procedure uses the fragmentation estimation query to determine if the indexes are fragmented, before running the optimization. It will therefore only optimize the indexes if they are fragmented.

One method for checking whether the indexes are fragmented involves counting the number of rows for each term in the DRxxxK table:

```
SELECT AVG(COUNT(*)) FROM DR$index_name$I
   GROUP BY TOKEN_TEXT HAVING COUNT(*) > 1;
```

Note: Ignore all words with only a single row in the index table.

A value greater than 10 from this query may indicate the need to optimize the index, but experimentation should yield the best value in any particular circumstances. Very large tables will inevitably have a lot of rows where the TOKEN_INFO data overflows the 4K internal limit, so you would expect the average to be greater on large systems.

See also: The Oracle Text Performance FAQ at:
`http://otn.oracle.com/products/text/`

7.5 Setting up Oracle Text Searching

There are four main steps for setting up Oracle Text in Oracle9*i*AS Portal:

- Step 1: Set up the Global Page Settings
- Step 2: Creating the Oracle Text Indexes
- Step 3: Enable and Configure Oracle9iAS Portal Text Searching
- Step 4: Maintain an Oracle9iAS Portal Text Index

7.5.1 Step 1: Set up the Global Page Settings

The first step requires you to configure the global page settings in the following way:

1. In the **Services** portlet, go to the Proxy Settings page. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's **Administer** tab.

Figure 7–2 Global Page Settings proxy server

2. In the **Add Proxy Server** section:
   - Enter the host name of your proxy server for the HTTP Server. Make sure not to prefix `http://` to the proxy server name.
3. In the **No Proxy Server Setting** section:
   - Enter the domains that you do not want redirected to the proxy server.
4. Click **OK**.

7.5.2 Step 2: Creating the Oracle Text Indexes

By default Oracle9*i*AS Portal text indexes are built at install time. You will only need to build a new index in case the index creation failed during installation or if the indexes have since been dropped. You can tell that indexes have been created by going to the Oracle9*i*AS Portal home page's **Administer** tab. In the **Services** portlet,

click **Search Settings.** If the button caption in the **Specify Oracle Text Search Properties**. portlet reads 'Drop Index', it means that the index already exists and that it does not need to be created. Otherwise the button caption will read 'Create Index'.

You can create the Oracle Text indexes as follows:

1. Navigate to the *ORACLE_HOME/portal/src/wws* directory.
2. In SQL*Plus Log on using the user name and password for the schema that owns the Oracle9*i*AS Portal page group. For example, if the schema name is "SCOTT", log on with the user name "SCOTT" and the appropriate password.
3. In SQL*Plus type the following command:

   ```
   ctxcrind.sql
   ```

Alternatively, you can go to the Oracle9*i*AS Portal home page's Administer tab. In the Services portlet, click Search Settings. In the **Specify Oracle Text Search Properties** section, click **Create Index**.

- If properly run, the message "Index created" appears and the Oracle Text indexes are created on the server.
- If you fail to create the index, check that your system has met all the requirements in Section 7.2, "Prerequisites".

Note: Since Index creation can take quite a while, there is a good chance that your browser will time out during the operation. If mod plsql tries to clean up the session after the timeout the index creation may be rolled back. It is therefore recommended to perform the index creation in SQL*Plus as described above.

The following Oracle Text indexes are created:

Table 7–1 Oracle Text indexes created

Index name	Description
WWSBR_CORNER_CTX_INDX	Index for all page attributes.
WWSBR_THINGS_CTX_INDX	Index for all item attributes.

Table 7–1 Oracle Text indexes created

Index name	Description
WWSBR_PERSP_CTX_INDX	Index for all Perspective attributes.
WWSBR_DOC_CTX_INDX	Index for all item and page document content data. (Only items and pages can have attached documents.)
WWSBR_TOPIC_CTX_INDX	Index for all category attributes.
WWSBR_URL_CTX_INDX	Index for all item and page URL content data.

Note: The time required for creating indexes varies depending on the number of items you have in your page group.

In troubleshooting, you may need to reinstall Oracle Text, or you may need to recreate the `ctxsys` schema. In both of these cases, you need to run the following script in SQL*Plus to reset the Oracle9*i*AS Portal Oracle Text environment:

```
inctxgrn.sql
```

This file is located in the `ORACLE_HOME/portal/src/wws` directory.

Log on using the user name and password for the schema that owns the Oracle9*i*AS Portal page group. For example, if the schema name is "SCOTT", log on with the user name "SCOTT" and the appropriate password.

7.5.3 Step 3: Enable and Configure Oracle9*i*AS Portal Text Searching

You can enable and configure the Oracle Text settings in Oracle9*i*AS Portal in the following way:

1. In the **Services** portlet, click **Search Settings**. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's **Administer** tab. (See Figure 7–3, "Services portlet Oracle Text properties")

Note: If you see the message, "Oracle Text is not installed", Oracle Text was not installed with the database and is not available for your page groups. Arrange with your database administrator to have Oracle Text installed. After it is installed, you need to run the following command in SQL*Plus:

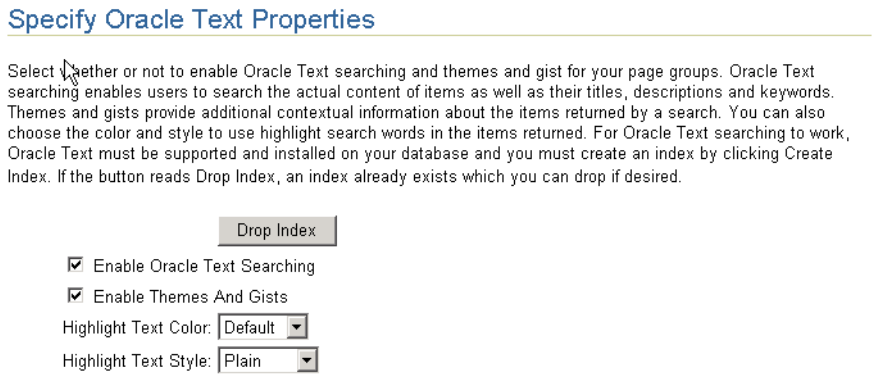
```
inctxgrn.sql
```

This file is located in the `ORACLE_HOME/portal/src/wws` directory.

Log on using the user name and password for the schema that owns the Oracle9*i*AS Portal page group. For example, if the schema name is "SCOTT", log on with the user name "SCOTT" and the appropriate password.

2. Select **Enable Oracle Text Searching** to make Oracle Text searching available in your page groups.

Figure 7–3 Services portlet Oracle Text properties



3. Select **Enable Themes And Gists** to create a theme and gist for each item returned by the search.
   - A theme shows the nouns and verbs that occur most frequently within the item.

- A gist displays a brief summary of the item, derived from how frequently those nouns and verbs appear.

> **Note:** Themes and Gists are not available for all languages.

4. In the **Highlight Text Color** list, choose the color to highlight the search words in the HTML renditions of the items returned by the search.
5. In the **Highlight Text Style** list, choose the style to apply to the search words in the HTML renditions of the items returned by the search.
6. Click **OK**.

7.5.4 Step 4: Maintain an Oracle9*i*AS Portal Text Index

Oracle Text lets you create a text index (an inverted index) on documents stored in the database. Updating an inverted index requires heavy processing, so changes to a text column are queued and processed in batch. The process of updating the inverted index based on the queue is referred to as "synchronizing" the index.

The second aspect of maintaining your Oracle Text index is optimizing. As your index is synchronized, it grows in such a way as to consume more disk space than necessary and reduces the efficiency of queries.

Optimizing your index works differently depending on the mode you select. Optimizing in FAST MODE works on the entire index and compacts fragmented rows, but does not remove old data. FULL MODE permits optimization of the whole index or a portion of the index and both compacts fragmented rows and removes old data.

> **See also:** Oracle Text documentation for the ALTER INDEX command.

Oracle Text gives you full control over how often each text index is synchronized. You can choose to synchronize every five seconds, for example, if it is important for your application to reflect text changes quickly in the index. You can choose to synchronize once a day, for more efficient use of computing resources and a more optimal index.

After creating your Oracle Text index, you need to consider a strategy for maintaining the index. For example, if you have many inserts, updates, or deletes throughout the day, consider synchronizing the Oracle Text index on a daily basis.

Oracle9*i*AS Portal provides new procedures and scripts for updating and optimizing the six Oracle9*i*AS Portal Text indexes.

7.5.4.1 Synchronizing Oracle9*i*AS Portal Text Indexes

A new Oracle9*i*AS Portal procedure for Index synchronization has been added that can be called to synchronize all of the relevant indexes:

```
wwv_context.sync();
```

This call procedure uses the `ctx_ddl.sync_index` procedure and can be executed by the Portal schema owner from SQL*Plus, by using the command

```
exec wwv_context.sync();
```

This procedure will index those rows that are out of date. To keep the indexes synchronized a job can be set up to call this procedure. The script `portal/src/wws/textjsub.sql` is provided for this.

You can also index via the following command:

```
ALTER INDEX indexname REBUILD ONLINE PARAMETERS('SYNC')
```

Alternatively, you can still use the procedure:

```
ctx_ddl.sync_index
```

Note: The Context Server (ctxsrv) and the use of `ctx_schedule` has been deprecated and should no longer be used.

Note also that this procedure needs to be run for all the indexes.

The view *ctx_user_pending* can be used to see how many rows there are for each of the user's indexes that need indexing. Note that the script `wws/textstat.sql` gives a number of details about the state of the Portal Text indexes and associated synchronization jobs. One of these details is the number of rows from each of the Portal Text indexes that are pending.

It is more efficient to synchronize a larger number of rows on a single occasion than to repeatedly synchronize a smaller number of rows. However, indexing a larger number of rows at once places a heavier load on the server. Synchronizing more frequently will increase the total amount of work done will spread the load on the server.

7.5.4.2 Optimizing the Oracle Text Indexes

Optimizing Indexes prevents fragmentation, which can cause performance loss. To optimize the Oracle Text indexes you can use the new Oracle9*i*AS Portal procedure:

```
wwv_context.optimize();
```

This procedure calls *ctx_ddl.optimize_index* for each of the Oracle9*i*AS Portal indexes and can be executed by the Portal schema owner from SQL*Plus, by using the command

```
exec wwv_context.optimize();
```

To keep the indexes optimized a job can be set up to call this procedure. The script `portal/src/wws/optjsub.sql` is provided for this.

Alternatively, you can call the *ctx_ddl.optimize_index* procedure directly from SQL*Plus.

Note: The Context Server (ctxsrv) and the use of *ctx_schedule* has been deprecated and should no longer be used.

Note also that this procedure needs to be run for all the indexes.

7.5.4.3 Stopping the Index Maintenance

You can stop the text synchronization job by logging on as the Portal schema owner and running the script `textjsub.sql` pass in STOP as the first argument. The second two arguments are then ignored. This will stop the synchronization job.

7.6 Dropping an Oracle Text Index

Dropping an index is a very time-consuming and resource-intensive operation so plan this task during non-business hours.

You would drop an Oracle Text index in the following situations:

- You want to disable Oracle Text and switch back to a basic and advanced search only.
- You know that a significant amount of new content has been added to your page group. First, drop your index, then, recreate your index.

You can drop the Oracle Text indexes as follows:

1. Navigate to the `ORACLE_HOME/portal/src/wws` directory.

2. In SQL*Plus Log on using the user name and password for the schema that owns the Oracle9*i*AS Portal page group. For example, if the schema name is "SCOTT", log on with the user name "SCOTT" and the appropriate password.
3. In SQL*Plus type the following command:

   ```
   ctxdrind.sql
   ```

> **Note:** When the Portal Text indexes are dropped, any views and packages that reference tables on which the indexes were created will become invalid.
>
> These views and packages will be automatically revalidated when they are next accessed. Alternatively, it is possible to revalidate the views and packages manually.
>
> The script oracle_home/rdbms/admin/utlrp.sql can be used to revalidate all of the packages and views. It should be run as the portal schema owner and is supplied with the database.

Alternatively, you can go to the Oracle9*i*AS Portal home page's Administer tab. In the Services portlet, click Search Settings. In the **Specify Oracle Text Search Properties** section, click **Drop Index**.

- If properly run, the message "Index dropped" appears and the Oracle Text index is dropped on the server.

> **Note:** Since dropping the Indexes can take quite a while, there is a good chance that your browser will time out during the operation. If mod plsql tries to clean up the session after the timeout the index drop may be rolled back. It is therefore recommended to perform the index creation in SQL*Plus as described above.

7.7 Multilingual Functionality (Multilexer)

> **Note:** You may need to increase your tablespace to at least 20 MB to support Multilexer.

Multilexer allows you to use language-specific features on documents of different languages stored in the same table. Multilexer is a feature of the index and is configured during index creation. Multilexer requires an extra column in your table, which identifies the language of each document.

At query time, the Multilexer chooses a language-specific lexer to lex the query tokens. This is based on the NLS_LANG setting for the query session. Thus, a query session in the FRENCH language uses the lexer for FRENCH.

During installation of Oracle9*i*AS Portal, the `sbrimtlx.sql` script creates the language-specific lexer preferences and gathers them under a single multilexer preference.

7.8 Oracle Text-related Procedures Created in Oracle9*i*AS Portal

The Oracle9*i*AS Portal installation creates the following procedures in the `ctxsys` schema. These procedures are created to support the user datastores that are used in page groups for Oracle Text indexing.

- `WWSBR_CORNER_CTX_`*n*
- `WWSBR_DOC_CTX_`*n*
- `WWSBR_PERSP_CTX_`*n*
- `WWSBR_THING_CTX_`*n*
- `WWSBR_TOPIC_CTX_`*n*

where *n* is the `user_id` of the Oracle9*i*AS Portal schema which may be different for each database. This value is the `user_id` column value from `all_users`.

7.9 Oracle Ultra Search

This section provides information about Oracle Ultra Search and on how to perform the required database and middle tier configuration. Specific topics in this section include:

Oracle Ultra Search Overview

Configuring the Oracle9i Application Server Infrastructure

Configuring the Oracle9i database for Oracle Ultra Search

Configuring the Oracle Ultra Search Middle Tier Component

Configuring Remote Crawler Hosts

The Oracle Ultra Search Portlet Sample

7.9.1 Oracle Ultra Search Overview

In this Oracle Ultra Search overview section we will cover the following topics:

About Oracle Ultra Search

About the Oracle Ultra Search Sample Query Applications

About the Oracle Ultra Search Administration Tool

7.9.1.1 About Oracle Ultra Search

Oracle Ultra Search lets you index and search Web sites, database tables, files, mailing lists, Oracle9*i*AS Portal, user-defined data sources. As such, you can use Oracle Ultra Search to build different kinds of search applications. Oracle Ultra Search has the following components:

- Server component
- Administration tool
- Crawler
- Query API
- Email API
- Query applications

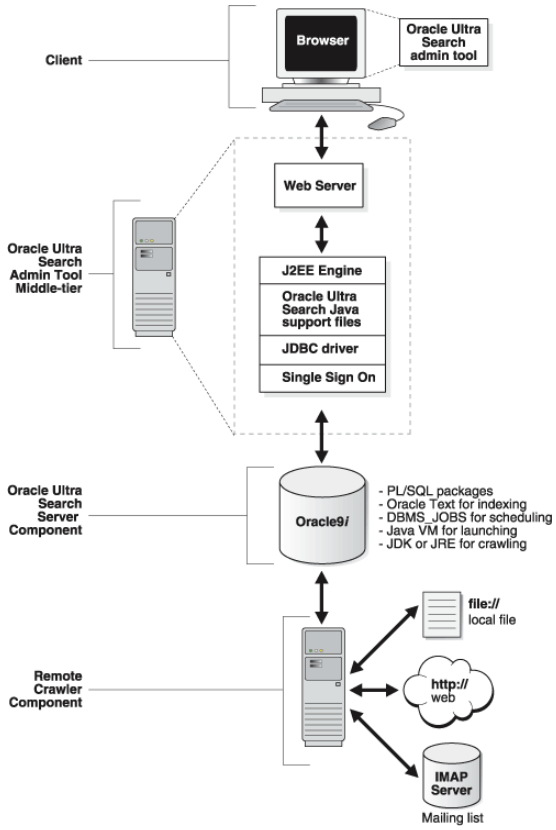
See also:

- The complete Oracle Ultra Search documentation is located at: `ORACLE_HOME/ultrasearch/doc/help/toc.htm`
- Visit `http://otn.oracle.com/products/ultrasearch` for Oracle Ultra Search white papers and presentations.

Oracle Ultra Search is integrated with Oracle9*i*AS Portal. This allows Oracle9*i*AS Portal users to add a powerful multi-repository search to their portal pages. It also has the capability to crawl Oracle9*i*AS Portal's own repository and make it searchable.

The following image shows an overview of the Oracle Ultra Search architecture:

Figure 7–4 Oracle Ultra Search architecture



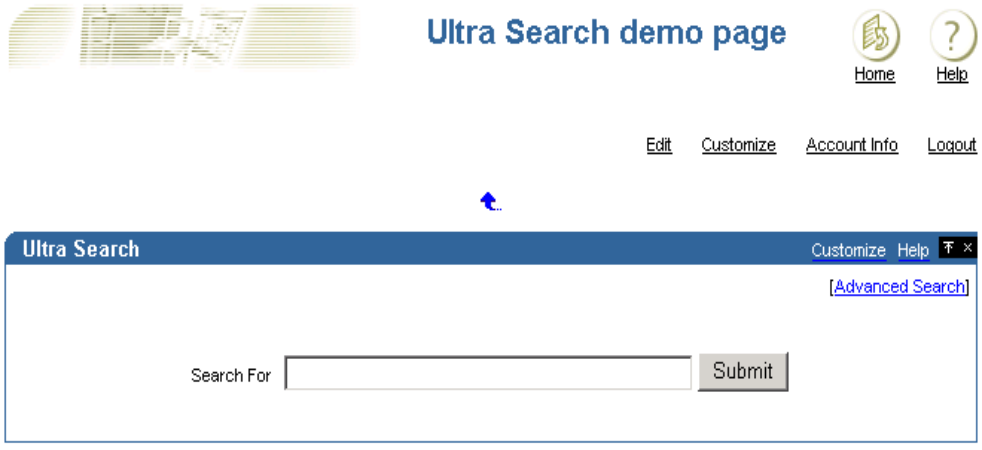
See also: For a complete overview and detailed information about Oracle Ultra Search, refer to the Oracle Ultra Search online Help.

7.9.1.2 About the Oracle Ultra Search Sample Query Applications

Oracle Ultra Search includes fully functional sample query applications to query and display search results. The query applications are written as JavaServer Page (JSP) applications.

The sample query applications also include a sample search portlet as shown in the following image.

Figure 7–5 Oracle Ultra Search portlet



The sample Oracle Ultra Search portlet demonstrates how to write a search portlet for use in Oracle9*i*AS Portal.

When the user issues a query in any of the query applications, a hit list containing query results is returned. The user can select a document to view from the hit list. A hit list can include HTML documents, files, database table content, archived e-mails, or other items. The Oracle Ultra Search sample query applications also incorporate an email browser for reading and browsing e-mails.

Figure 7–6 Example of query results in the Oracle Ultra Search portlet

If you do not want to use the Oracle Ultra Search sample query applications, you can build your own query application by directly invoking the Oracle Ultra Search Java Query API . Because the API is coded in Java, you can invoke the API methods from any Java-based application, such as from a Java servlet or a JavaServer page (as in the case of the provided sample query applications). For rendering e-mails that have been crawled and indexed, you can also directly invoke the Oracle Ultra Search Java Email API methods.

> **See also:** the Oracle Ultra Search online documentation for information about the Oracle Ultra Search Sample Query Applications, and the README file located at:
>
> `ORACLE_HOME/ultrasearch/sample/sample_readme.htm`

7.9.1.3 About the Oracle Ultra Search Administration Tool

The Oracle Ultra Search administration tool is a Web application for configuring and scheduling the Oracle Ultra Search crawler. It allows user management operations on either database users or SSO users. Authenticated SSO users never

see the Oracle Ultra Search login screen. Instead, they can immediately choose an Oracle Ultra Search instance.

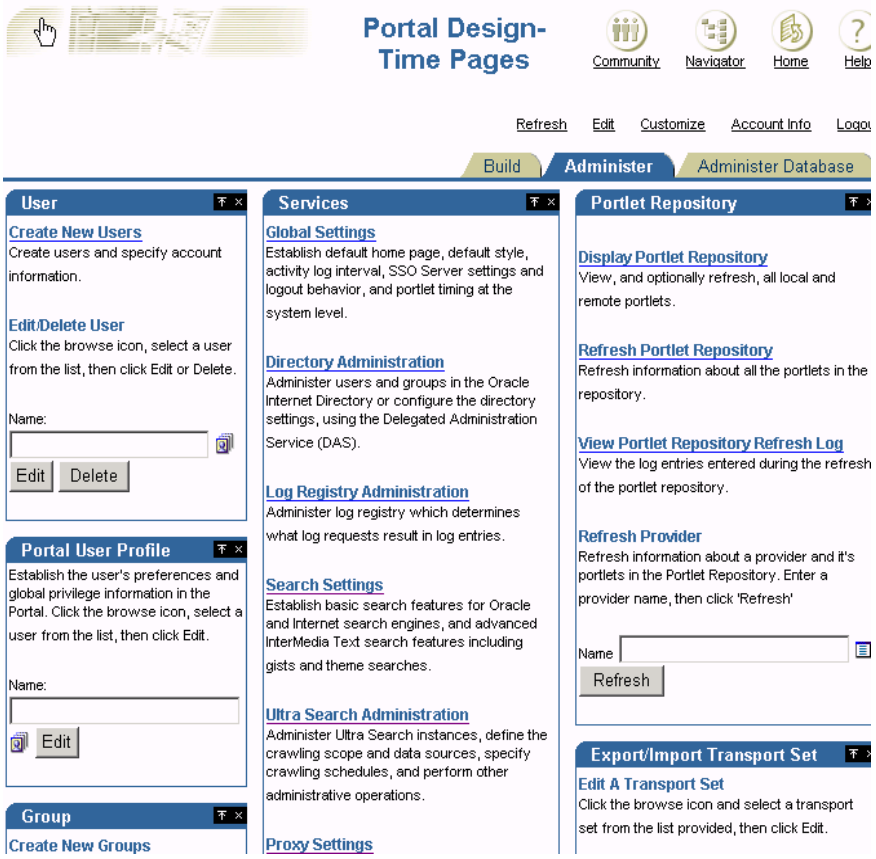
From the Oracle Ultra Search administration tool you can:

- Define Oracle Ultra Search instances
- Manage administrative users
- Define data sources and assign them to data groups
- Configure and schedule the Oracle Ultra Search crawler
- Set query options

The Oracle Ultra Search administration tool and the Oracle Ultra Search sample query applications are part of the Oracle Ultra Search middle tier components module. However, the Oracle Ultra Search administration tool is independent from the Oracle Ultra Search sample query applications. Therefore, they can be hosted on different machines to enhance security or scalability.

You can access the Oracle Ultra Search administrative interface through Oracle9*i*AS Portal. In the **Services** portlet, go to the **Ultra Search Administration** page. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's **Administer** tab (see Figure 7–7, "The Oracle Ultra Search administration portlet").

Figure 7–7 The Oracle Ultra Search administration portlet



See also: For more information about Oracle Ultra Search, or the Oracle Ultra Search administration tool, refer to the Oracle Ultra Search online Help.

7.9.2 Configuring the Oracle9*i* Application Server Infrastructure

The Oracle Ultra Search server tier will be installed with the Oracle9*i* Application Server infrastructure by default.

During the installation of Oracle9*i* Application Server infrastructure or the Oracle database server, the Oracle Ultra Search server component is installed. The following activity occurs during this process:

- All Oracle Ultra Search server component files are copied into a directory named "ultrasearch". This directory resides immediately under the ORACLE_HOME of the designated database installation.
- The database user WKSYS with password WKSYS is created. You should change this password later for security purposes. All Oracle Ultra Search database objects are installed in this user's schema.
- Various PL/SQL scripts are run against the database as user WKSYS. These scripts install and create various database objects.

Ensure that the following five environment variables are set whenever you operate on this Oracle instance. This is especially important when working on a host that contains multiple Oracle products (and hence multiple Oracle homes). The environment variables and their values are as follows:

1. `ORACLE_HOME`: The directory in which you have installed Oracle.
2. `ORACLE_SID`: The Oracle instance SID of the database that you specified during the installation process.
3. `PATH`: Must be extended to include the bin directory of the newly installed Oracle home (for example, ORACLE_HOME/bin:$PATH)
4. `TNS_ADMIN`: Must be set to the network/admin subdirectory in the newly installed Oracle home.
5. `LD_LIBRARY_PATH`: Must include all necessary system library directories.

Note: Your Oracle DBA can edit the `oraenv` script to ensure that all these environment variables are correctly set every time you begin a new shell. Alternatively, you can edit your shell startup script, such as the .cshrc or .bashrc file, in your home directory.

7.9.3 Configuring the Oracle9*i* database for Oracle Ultra Search

The operations described in this section are database administration operations. They can be performed using Oracle Enterprise Manager or SQL*Plus.

> **See also:** *Oracle9i Application Server Administrator's Guide* for more information on Oracle Enterprise Manager.

This section lists the necessary steps for:

- Configuring the Oracle Server for Oracle Ultra Search.
- Creating tablespaces and users for each Oracle Ultra Search instance.

Step 1: Tune the Oracle Database

Tuning the Oracle Database for Oracle Ultra Search consists of checking and increasing the size of your log files and increasing the size of the undo space.

a. Increase the size of the Oracle redo logs, if necessary.

Every instance of an Oracle database has an associated online redo log, which is a set of two or more online log files that record all committed changes made to the database. Online redo logs protect the database in the event of an instance failure. The size of redo log files determines the frequency of redo log file switches. This, in turn, significantly impacts text indexing speed. To reduce the frequency of logfile switches, ensure that the redo log files are each 10Mb or more.

See also: *Oracle9i Designing and Tuning for Performance*, or the *Oracle9i Application Server Administrator's Guide* for details on tuning your system.

b. Increase the size of the undo space.

Every Oracle database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. Oracle refers to these records collectively as undo. The undo space created by the Oracle Installer is likely to be too small.

Historically, Oracle has used rollback segments to store undo. Oracle now offers another method of storing undo that eliminates the complexities of managing rollback segment space, and enables DBAs to exert control over how long undo is retained before being overwritten. This method uses an undo tablespace.

Oracle Corporation recommends that you use automatic undo management and increase the undo space using an UNDO_TABLESPACE.

See also: *Oracle9i Application Server Administrator's Guide* for details on using automatic undo management.

Step 2: Create and Assign the Temporary Tablespace to the CTXSYS User

The starter database created by the Oracle Installer most likely creates a temporary tablespace that is too small. Oracle Ultra Search uses the Oracle Text engine intensively. Therefore, a large temporary tablespace must be created for the Oracle Text system user CTXSYS.

If you want greater read and write performance, create a raw tablespace.

When you have created the temporary tablespace, assign it as the temporary tablespace for the CTXSYS user. To do so, you must log on as the SYSTEM or SYS user. You can assign the temporary tablespace to the CTXSYS user with the following statement:

```
ALTER USER CTXSYS TEMPORARY TABLESPACE <NEW_TEMPORARY_TABLESPACE>
```

> **See also:** *Oracle9i Application Server Administrator's Guide* for information on how to create a temporary tablespace.

Step 3: Create a Tablespace for Each Oracle Ultra Search Instance User

For each Oracle Ultra Search instance, you must create a tablespace large enough to contain all data obtained during the crawling and indexing processes. This amount is naturally subject to the amount of data you intend to crawl and index. However, it is often not possible to know in advance how much data you intend to collect. Try to obtain an estimate of the cumulative size of all data you want to crawl.

If you cannot estimate the size, then try to allocate as much space as possible. If you run out of disk space later, Oracle Ultra Search is able to resume crawling after you have added more datafiles to the instance tablespace.

Pay special attention to the STORAGE clause in your CREATE TABLESPACE statement. The amount of data to be stored in the tablespace can potentially be very large. This can cause the Oracle Server to progressively allocate many new extents when more storage space is needed. If the extent management clause specifies that each new extent is to be larger than the previous extent (that is, the PCTINCREASE setting is nonzero), then you could encounter the situation where the next extent that the Oracle Server wants to allocate is larger than what is available. In such a situation, indexing is halted until new extents can be added to the tablespace.

To help mitigate this problem, certain instance-specific tables have explicit storage parameter settings. The initial extent size, next extent size, and PCTINCREASE setting are defined for these tables. These tables are created when a new instance is created. The tables and their storage clause settings are as follows:

`DR$WK$DOC_PATH_IDX$I`

(initial extent size 5M, next extent size 50M, PCTINCEASE 1)

`DR$WK$DOC_PATH_IDX$K`

(initial extent size 5M, next extent size 50M, PCTINCEASE 1)

See also: *Oracle9i SQL Reference* for information on creating tablespaces and managing storage settings.

If you want greater read and write performance, create raw tablespaces.

Note: Be sure to create a new large tablespace for each Oracle Ultra Search instance user.

Step 4: Create and Configure New Database Users for Each Oracle Ultra Search Instance

The Oracle Ultra Search system uses Oracle's Fine Grained Access Control feature to support multiple Oracle Ultra Search instances within one physical database. This feature is especially useful for large organizations or application service providers (ASPs) that want to host multiple disjoint search indexes within one physical installation of Oracle.

The Oracle Ultra Search system requires that each Oracle Ultra Search virtual instance belong to a unique database user. Therefore, as part of the installation process, you must create one or more new database users to own all data for your Oracle Ultra Search instance. (Note: If you intend to create more than one database instance, you should also create multiple user tablespaces - one for each user).

You need to grant certain roles and privileges to each Oracle Ultra Search user. For convenience, the WKUSER role has all the necessary privileges.

Enter the following statements to create and configure a new user. You can run these statements as the WKSYS, SYSTEM, or SYS database user.

```
CREATE USER <username> IDENTIFIED BY <password> DEFAULT TABLESPACE <default_tbs>
TEMPORARY TABLESPACE <temporary_tbs> QUOTA UNLIMITED ON <default_tbs>;
```

where:

Table 7–2 parameters for creating new users on an Oracle Ultra Search instance

parameter	description
username	Name of the Oracle Ultra Search instance owner
password	Password of the Oracle Ultra Search instance owner
default_tbs	Default tablespace for the Oracle Ultra Search instance created in step 3
temporary_tbs	Temporary tablespace created in step 2

```
GRANT WKUSER TO <username>;
```
After the above steps, WKSYS or an Oracle Ultra Search super-user, can create a Oracle Ultra Search instance on this user schema.

If you want this user to have the general administrative privilege or the super-user privilege of Oracle Ultra Search, you can log in as an Oracle Ultra Search super-user or WKSYS and click on the "Users" tab to grant the appropriate privilege.

Note: After the infrastructure database is installed, all the user schema passwords will be randomized. To login as user WKSYS, you can change the WKSYS schema password by running the following statement as the SYSTEM, or SYS database user.

```
ALTER USER WKSYS IDENTIFIED BY <password>;
```

Step 5: Gather Statistics for the Tables

If you notice performance degradation on the crawler, it might be because statistics have not been gathered for the tables. You should gather statistics for the following tables: `wk$url`, `wk$doc`, and `dr$wk$doc_path_idx$i`. Statistics for the `wk$url` table are the most important. You must regularly gather statistics, because statistics are used by the cost-based optimizer to generate the best execution plan. Make sure that the crawler is not running during the performance tuning period to avoid interference.

You can use the `DBMS_STATS PL/SQL` package or the `ANALYZE` procedure to gather statistics. The `DBMS_STATS` package can be run on either the table level or the schema level. Running on the schema level computes the statistics for the all the objects in the schema including the tables and indexes. Oracle Corporation recommends using the `DBMS_STATS` package.

Connect to the schema owning the Oracle Ultra Search instance. For example:

```
EXEC DBMS_STATS.GATHER_TABLE_STATS('<schema_name>', '<table_name>', null, DBMS_
STATS.AUTO_SAMPLE_SIZE);
```
or

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS('<schema_name>', DBMS_STATS.AUTO_SAMPLE_
SIZE);
```
or

```
ANALYZE TABLE <table_name> ESTIMATE STATISTICS SAMPLE 20 percent;
```
where `schema_name` is the owner of the Oracle Ultra Search instance and `table_name` is the table you want to gather statistics for (for example, `wk$url`).

Occasionally rebuilding the B-tree indexes can also improve performance by freeing up disk space. For example:

```
ALTER INDEX <index_name> REBUILD;
```

where index_name is the index that you want to rebuild.

To get a list of the indexes, run the following statement:

```
SELECT index_name FROM user_indexes WHEREindex_type='NORMAL';
```

7.9.4 Configuring the Oracle Ultra Search Middle Tier Component

Note: If you checked the "Oracle9*i*AS Portal" option on the "Component Configuration" Oracle Installer screen, then the configuration steps in the following section are automatically performed by Oracle9*i*AS Portal Configuration Assistant. If not, then you must manually perform the steps under *Configuring Oracle Ultra Search Middle Tier Component with Oracle HTTP Server and OC4J* in the Oracle Ultra Search online help to configure your existing Web server.

Editing the data-sources.xml File

The Oracle Ultra Search administration tool middle tier component and the Oracle Ultra Search Oracle9*i* Application Server query API use the data source functionality of the J2EE container. In order to function properly, the data-sources.xml file needs to be edited.

In the ORACLE_HOME/j2ee/OC4J_Portal/config directory, edit the file data-sources.xml. Under the <data-sources> tag add the following:

```
<data-source
   class="oracle.jdbc.pool.OracleConnectionCacheImpl"
   name="UltraSearchDS"
   location="jdbc/UltraSearchPooledDS"
   username="<username>"
   password="<password>"
   url="jdbc:oracle:thin:@<database_host>:<oracle_port>:<oracle_sid>"
/>
```

where

username and *password* are the Oracle Ultra Search instance owner's database username and password, *database_host* is the host name of the back end database machine, *oracle_port* is the port to the user's Oracle database, and *oracle_sid* is the SID of the user's Oracle database.

In addition to configuring the username, password, and JDBC URL, *data-sources.xml* also allows configuration of the connection cache size, as well as the cache scheme.

The following tag specifies the minimum and maximum limits of the cache size, the inactivity time-out interval, and the cache scheme.

```
<data-source
   class="oracle.jdbc.pool.OracleConnectionCacheImpl"
   name="UltraSearchDS"
   location="jdbc/UltraSearchPooledDS"
   username="wk_test"
   password="wk_test"
   url="jdbc:oracle:thin:@localhost:5521:isearch"
   min-connections="3"
   max-connections="30"
   inactivity-timeout="30" <property name="cacheScheme"
   value="DYNAMIC_SCHEME"/>
/>
```

There are three types of caching schemes:

- DYNAMIC_SCHEME
- FIXED_WAIT_SCHEME
- FIXED_RETURN_NULL_SCHEME

> **See also:** *Oracle9iAS Containers for J2EE Services Guide* for more information.

Editing the ultrasearch.properties File

The `ultrasearch.properties` file specifies which database the Web application and JSP application connect to. This file is located in the following directory:

`ORACLE_HOME/ultrasearch/webapp/config`

You must specify the hostname, port, and SID of the Oracle instance and listener. To do this, edit the line that begins with "*connection.url*" to read:

```
connection.url=jdbc:oracle:thin:<hostname>:<port>:<SID>
```

For *hostname*, enter the full host name of the Oracle9*i* server instance running Oracle Ultra Search. For *port*, enter the listener port number for the Oracle9*i* server instance. For *SID*, enter the Oracle9*i* server instance ID.

Here is an example connection.url string:

```
connection.url=jdbc:oracle:thin:@ultrasearch.us.oracle.com:152
1:myInstance
```

If you chose to configure the Oracle Ultra Search middle tier component with Oracle HTTP Server and OC4J, then you must also edit the line that begins with "admin.srchome" to read:

```
admin.srchome=<jsp_src_home>
```

On UNIX:

```
admin.srchome=ORACLE_HOME/ultrasearch/webapp/isearch_admin
```

On Windows NT/2000:

```
admin.srchome=ORACLE_HOME\ultrasearch\webapp\isearch_admin
```

This is the location of the Oracle Ultra Search administration tool JSP pages.

Note: You should not need to change the first line, because it is the name of the Oracle JDBC driver.

Starting the Web Server

You should start the Web server using the Oracle Enterprise Manager console.

> **See also:** *Oracle9i Application Server Administrator's Guide* for more information on Oracle Enterprise Manager and using the Oracle Enterprise Manager console.

To test if the Oracle HTTP server is started, visit the Oracle Ultra Search welcome page:

```
http://hostname.domainname:HTTPport/ultrasearch/index.html
```

This page provides general information about Oracle Ultra Search, and it also contains links to the Oracle Ultra Search administration tool, as well as Oracle Ultra Search sample query JSP page.

```
If you deploy the Oracle Ultra Search middle tier with OC4J, start OC4J by
invoking java -jar ORACLE_HOME/j2ee/home/oc4j.jar -config ORACLE_HOME/j2ee/OC4J_
Portal/config/server.xml
```

OC4J can be configured to start automatically when the Oracle HTTP server starts.

Testing the Oracle Ultra Search Administration Tool

You can test your changes by attempting to log on to the Oracle Ultra Search administration tool at:

```
http://hostname.domainname:HTTPport/ultrasearch/admin/index.jsp
```

Where hostname.domainname is the full name of the host where you have just installed the Oracle Ultra Search middle tier component, and HTTPport is the default Web server port. If you are running the Web browser on the same host, you can enter "localhost".

During the installation of the Oracle Ultra Search server component, you should have created a new Oracle Ultra Search instance owner. The instance owner is created in Step 4 of the Oracle Ultra Search server component installation process. Log on to the Oracle Ultra Search administration tool by entering the Oracle Ultra Search instance owner's database username and password.

If you log on to the Oracle Ultra Search administration tool successfully, then you have completed the Oracle Ultra Search administration tool configuration process.

You can also access the Oracle Ultra Search administrative interface through Oracle9*i*AS Portal. In the **Services** portlet, go to the **Ultra Search Administration** page. By default, the Services portlet is located on the Oracle9*i*AS Portal home page's **Administer** tab (see Figure 7–7, "The Oracle Ultra Search administration portlet").

Testing the Oracle Ultra Search JSP Sample Query Applications

After you verify that the Oracle Ultra Search administration tool is working, you should be able to run the Oracle Ultra Search JSP sample query applications. You can access the sample query source code by going to the directories list. You can also see a working demo of each sample query JSP page with the URL root, and you can append the correct JSP file name at the end of the URL root.

The following is a list of URLs and locations of various sample application related files:

Table 7–3 Sample Query File Locations

Description	Path or URL
Test link for the Oracle Ultra Search JSP sample query applications	`http://hostname.domainname:HTTPport/ultrasearch/query/search.jsp`
Root query directory	`ORACLE_HOME/ultrasearch/sample/query/`
URL root for the query	`http://hostname.domainname:HTTPport/ultrasearch/query/`

Table 7–3 Sample Query File Locations

Description	Path or URL
9*i*AS query (the query sample JSP pages that use the 9*i*AS query API and includes the files usearch.jsp and search.jsp)	`ORACLE_HOME/ultrasearch/sample/query/`
URL root for the 9*i*AS query	`http://hostname.domainname:HTTPport/ultrasearch/query/`
9*i* query (query JSP that uses the 9*i* query API and includes gsearch.jsp)	`ORACLE_HOME/ultrasearch/sample/query/9i/`
URL root for the 9*i* query	`http://hostname.domainname:HTTPport/ultrasearch/query/9i/`
Oracle Ultra Search Portlet	`ORACLE_HOME/ultrasearch/sample/query/portlet/`
URL root for Oracle Ultra Search Portlet	`http://hostname.domainname:HTTPport/ultrasearch/query/portlet/`
Oracle Ultra Search Taglib	`ORACLE_HOME/ultrasearch/sample/query/tag/`
URL root for the Oracle Ultra Search Taglib	`http://hostname.domainname:HTTPport/ultrasearch/query/tag/`

7.9.5 Configuring Remote Crawler Hosts

The Oracle Ultra Search remote crawler functionality allows multiple crawlers to run in parallel on different hosts. All remote crawler hosts must share common resources, such as common directories and a common Oracle Ultra Search database.

> **See also:** Oracle Ultra Search online Help topic *Installing the Oracle Ultra Search Server Tier Component on Remote Crawler Hosts.*

7.9.6 The Oracle Ultra Search Portlet Sample

Oracle Ultra Search provides a search portlet that can be embedded in Oracle9*i*AS Portal pages. It is implemented as a JavaServer Page (JSP) application and called the Oracle Ultra Search Portlet Sample. The Oracle Ultra Search Portlet Sample is a web application that complies to the Oracle9*i*AS Portal portlet interface. By complying to the portlet interface, Oracle9*i*AS Portal users can create pages and embed Oracle Ultra Search portlets within those pages.

> **See also:** `http://portalcenter.oracle.com` for information about the Oracle9*i*AS Portal Developer Kit and information about the Oracle9*i*AS Portal portlet interface.

This portlet sample implements a provider that contains exactly one portlet. The provider name is simply "Ultra Search". The Oracle Ultra Search provider will belong to the "iAS Providers" provider group. The portlet contained within the Ultra Search provider is also called "Ultra Search".

Note that web providers are not registered as part of the Oracle9*i* Application Server install due to the fact that the provider must be up and running at the time that registration is performed. This is not possible, since the very last step performed during the installation is the starting of OC4J.

All web providers are, however, included in one of the following two provider groups:

- Oracle9*i*AS Providers
- PDK-Java Sample Providers

The Ultra Search provider is included in the Oracle9*i*AS Providers provider group. These groups will show up under the "Providers" tab in the Navigator. To register a provider, you need to follow these steps:

1. Open a provider group and click on the ***register*** link. This will take you to the provider registration screen where all the fields will be pre-populated with the correct information.
2. Click "OK".

Searching public data

The Oracle Ultra Search portlet enables Portal users to include Oracle Ultra Search's functionality in portal pages. However, it should be remembered that Oracle Ultra Search does not support any security model for search end-users. This means that all data crawled and indexed by Oracle Ultra Search is accessible to all users of a particular Oracle Ultra Search instance. There is no way to specify that a particular portal user has access to a subset of search results returned by Oracle Ultra Search.

Connecting to an Oracle Ultra Search instance

Oracle Ultra Search supports the creation of multiple Oracle Ultra Search instances. Each Oracle Ultra Search instance contains its own distinct index that can be queried against by the Oracle Ultra Search portlet. Each Oracle Ultra Search index requires its own database schema. The Oracle Ultra Search portlet must be

configured to query against a specific Oracle Ultra Search instance schema. This is done by configuring the ORACLE_HOME/j2ee/home/config/data-sources.xml file as follows:

```
<data-source
               class="oracle.jdbc.pool.OracleConnectionCacheImpl"
               name="UltraSearchDS"
               location="jdbc/UltraSearchPooledDS"
               username="ultrasearch_instance_schema"
               password="ultrasearch_instance_schema_password"
               url="jdbc:oracle:thin:@hostname:port:sid"
/>
```

where:

Table 7–4 Oracle Ultra Search connection parameters

Parameter	Description
ultrasearch_instance_schema	password of the schema
hostname	Oracle Ultra Search database hostname
port	Oracle Ultra Search database listener port
sid	Oracle Ultra Search database instance identifier

Note that the Sample Portlet shares the same data source entry as the Complete Sample Application.

Restrictions

Oracle9*i*AS Portal users should only embed Oracle Ultra Search portlets that are hosted on the same OC4J instance as Oracle9*i*AS Portal.

If Oracle9*i*AS Portal is installed on host A, Oracle Ultra Search will also be installed on host A. The Oracle Ultra Search provider will therefore also be hosted as a Web application on host A.

It is possible that the Oracle Ultra Search provider running on host A could be registered with a second Oracle9*i*AS Portal instance running on host B. However, if the Oracle Ultra Search portlet hosted on A is embedded within pages created in Portal B, the pop-up list-of-values will not work correctly. This is because of an security bug inherent in Javascript.

Portal pages created within Portal A should **only** embed the Oracle Ultra Search portlet from the provider running on host A and not from host B or any other host.

Portlet Sample Files

The Portlet sample files are located in the file

`ORACLE_HOME/ultrasearch/sample.ear`
The contents of that file are expanded into the directory

`ORACLE_HOME/ultrasearch/sample/query`
when the `sample.ear` file is first deployed by the application server. You can directly view the source code using your preferred text editor.

> **See also:** The file `ORACLE_HOME/ultrasearch/sample/query/portlet/README.html` for a complete list and descriptions of all the files used by the Portlet Sample, as well as a full description of how the portlet sample works.

8

Troubleshooting Oracle9*i*AS Portal

This chapter lists possible causes and solutions to errors that you may encounter while installing or using Oracle9*i*AS Portal.

> **See also:** For the most up-to-date troubleshooting information visit:
>
> http://portalcenter.oracle.com

Specific topics covered in this chapter include:

- Oracle9iAS Web Cache related issues
- Miscellaneous Issues

8.1 Oracle9*i*AS Web Cache related issues

There are a variety of possible issues that can come up when using Oracle9*i*AS Web Cache, or even after you disable the use of Oracle9*i*AS Web Cache. Specific issues are:

- Error: General invalidation message processing exception: ORA-06502: PL/SQL: numeric or value error (WWC-40018)
- Error: Could not open web cache connection. The portal use-web-cache setting is set to "on" while web cache may be down. (WWC-40019)
- Error: Error message "No Response from Application Server" displays in the browser
- Error: Intermittent error message "No Response from Application Server" displays in the browser

Error: General invalidation message processing exception: ORA-06502: PL/SQL: numeric or value error (WWC-40018)

Cause

This error can occur if a template is used by more than 25 page, and is caused by a known exception in the Oracle9*i*AS Web Cache invalidation message processing.

Solution

The workaround is to turn "off" Oracle9*i*AS Web Cache in the Portal. refer to Section 6.8.6, "Disabling Oracle9iAS Web Cache" for details on how to do turn off Oracle9*i*AS Web Cache.

Error: Could not open web cache connection. The portal use-web-cache setting is set to "on" while web cache may be down. (WWC-40019)

Cause

This error message is shown in the browser when the invalidation fails. In this case Oracle9*i*AS Portal was not able to send the invalidation message to Oracle9*i*AS Web Cache. This happens when the use of Oracle9*i*AS Web Cache is enabled in Oracle9*i*AS Portal and any of the following also happen either alone, or together:

- Oracle9*i*AS Web Cache might not be running. It might have been stopped purposefully in a conscious effort to front-end with the Oracle HTTP server, instead of Oracle9*i*AS Web Cache. Oracle9*i*AS Web Cache might have crashed, or it could have been bounced at the time when the invalidation messages were sent from Oracle9*i*AS Portal or Providers.

- The Oracle9*i*AS Web Cache invalidation port could have problems, this can happen because the Oracle9*i*AS Web Cache invalidation port value in Oracle9*i*AS Portal is different from the actual Oracle9*i*AS Web Cache invalidation port. The invalidation port value supplied to the portal using the "-cport_i" parameter during installation could also have been wrong, or the Oracle9*i*AS Web Cache invalidation port was changed in Oracle9*i*AS Web Cache after the Oracle9*i*AS Portal install.
- Some other process grabs the Oracle9*i*AS Web Cache invalidation port before Oracle9*i*AS Web Cache can listen on it.
- Trouble with Oracle9*i*AS Web Cache invalidation password, perhaps because the Oracle9*i*AS Web Cache invalidator password value in the Portal is different from the actual Oracle9*i*AS Web Cache invalidator password, or because the invalidator password value supplied to Oracle9*i*AS Portal using the "-wc_inv_pwd" parameter during the installation was wrong.
- Some security setting in Oracle9*i*AS Web Cache could have been enabled which might require the Oracle9*i*AS Web Cache administrator password, instead of the invalidator password.
- Trouble with Oracle9*i*AS Web Cache hostname because the Oracle9*i*AS Web Cache hostname value in the Portal is different from the actual Oracle9*i*AS Web Cache hostname, or the hostname value supplied to the portal using the "-host" and "-chost" parameter during the install could have been wrong.
- Oracle9*i*AS Web Cache might have been moved to a different Server after the Portal Install, leaving the host on which Oracle9*i*AS Web Cache runs inaccessible from Oracle9*i*AS Portal.
- The hostname specified may not include the entire domain name.
- Oracle9*i*AS Web Cache is not properly disabled. Oracle9*i*AS Portal can work even without Oracle9*i*AS Web Cache front-ending the middle-tier. To do this the "enable_wc_caching" flag in Oracle9*i*AS Portal should be turned "Off". If this is not done any operation which attempts to send an invalidation message will fail with exceptions like "Portal Web Cache settings is On while Web Cache may be down".
- A template is used by more than 25 page, and is caused by a known exception in the Oracle9*i*AS Web Cache invalidation message processing.

Solution

Check whether any of the above cases are causing this behavior.

If a template is used by more than 25 pages, the workaround is to turn "off" Oracle9*i*AS Web Cache in the Portal. refer to Section 6.8.6, "Disabling Oracle9iAS Web Cache" for details on how to do turn off Oracle9*i*AS Web Cache.

Error: Error message "No Response from Application Server" displays in the browser

Cause

The Oracle9*i* Application Server might be down.

Solution

Check if the Oracle9*i* Application Server needs to be started.

Error: Intermittent error message "No Response from Application Server" displays in the browser

Cause

- The Oracle9*i* Application Server might be bounced intermittently.
- The Network Connectivity to the Oracle9*i* Application Server might be poor.
- Oracle9*i*AS Web Cache might not have sufficient connections to the Oracle9*i* Application Server in order to handle all the incoming requests.

Solution

Check with the Oracle9*i* Application Server administrator if any of the above problems could be the cause of the problem.

Problem: Users see stale content after the operation which is expected to show the changed content, without an error messages in the browser.

Cause

- The invalidation could be an soft invalidation, which only takes effect after the invalidation job gets executed.
- An inappropriate invalidation API is used to invalidate content.
- The invalidation port specified in Oracle9*i*AS Portal could be pointing to the invalidation port of another Oracle9*i*AS Web Cache instance.
- Oracle9*i*AS Web Cache fails to process the invalidation message because it is under a heavy load.

- Oracle9*i*AS Web Cache invalidation message processing fails because of an invalid invalidation message syntax.
- Oracle9*i*AS Web Cache fails to invalidate cache contents despite receiving a correct invalidation message.
- Delayed Cache invalidation by Oracle9*i*AS Web Cache.

Solution

Check whether any of the above cases are causing this behavior.

Problem: There appears to be a delay in cached objects being invalidated

Cause

For example, a user customizes a portlet but does not see the customization, however, the change displays after a few seconds, when the browser is refreshed.

- The invalidation could be a *soft* invalidation which will only take effect after the invalidation job gets executed.
- Delayed cache invalidation by Oracle9*i*AS Web Cache.

Solution

Check whether any of the above cases are causing this behavior.

Problem: Nothing gets cached in Oracle9*i*AS Web Cache.

Cause

This happens when Oracle9*i*AS Web Cache caching is NOT enabled in the Portal. This could be because of the following reasons:

- The Oracle9*i*AS Web Cache "on", or "off" flag value supplied to Oracle9*i*AS Portal using the "-wc" parameter during the install was "off".
- The Oracle9*i*AS Web Cache "on", or "off" flag value could have been set to "off" by running some script in Oracle9*i*AS Portal, like, for example, the `cachseed.sql` script.
- The "Enable Web Cache Caching" flag in Oracle9*i*AS Portal has been set to false using the Oracle9*i*AS Web Cache administrative portlet in the Oracle9*i*AS Portal Administer tab of the services portlet.

- The File Cache was not cleared after turning "on" the "Enable Web Cache Caching" flag in the Portal.

Solution

This can be corrected by doing a MIDTIER install using Oracle9*i*AS Portal Configuration Assistant and specifying the value "on" for the -wc switch.

See also: Section B.2.2, "Using cachset.sql".

Problem: Objects get cached in Oracle9*i*AS Web Cache while the use of Oracle9*i*AS Web Cache has been set to "Off" in Oracle9*i*AS Portal

Cause

- The file cache might not have been cleared after turning "*off*" the "Enable Web Cache Caching" flag in the Oracle9*i*AS Portal.
- These may be non-portal objects which get cached because of the default cacheability rules in Oracle9*i*AS Web Cache.
- This can happen in the case of Oracle9*i*AS Portal objects which do not check the "enable_wc_caching" flag before setting the Oracle9*i*AS Web Cache specific cache headers.

Solution

- Clear the file cache.
- Check the cacheability rules in Oracle9*i*AS Web Cache.

Problem: Unable to navigate to the Web Cache Admin page from Oracle9*i*AS Portal

Cause

- The Oracle9*i*AS Web Cache Administration port value in Oracle9*i*AS Portal is different from the actual Oracle9*i*AS Web Cache Administration port.
- The Administration port value supplied to the portal using the "-cport_a" parameter during the install could be wrong.
- The Oracle9*i*AS Web Cache Administration port could have been changed in Oracle9*i*AS Web Cache after the Portal Install.

Solution

Rerun the OPCA in MIDTIER mode, or run the `cachseed.sql` script with the correct value for `-cport_a`.

Problem: Oracle9*i*AS Web Cache does not start

Cause

The Oracle9*i*AS Web Cache configuration files might be corrupted, or some other process might be listening in any of the ports Oracle9*i*AS Web Cache intends to listen.

Solution

Check if the configuration files are not corrupt, and see if there are any processes using the Oracle9*i*AS Web Cache port.

8.2 Miscellaneous Issues

Issues covered in this section include:

Problem: Unable to create Oracle Text indexes

Cause

If you encounter any of the following errors while creating an Oracle Text index a problem creating Oracle Text indexes occurred:

- Cannot grant CTXAPP Role to portal.
- ERROR: Creating datastore procedures in CTXSYS.
- ERROR: Setting up Oracle Text data stores.
- An unexpected error has occurred (WWS-32100)

Solution

Your system should meet all the requirements described in Section 1.1.7, "Oracle Text Requirements". Oracle Text must be installed in the same Oracle Home as the database.

You must disable connection pooling from the Database Access Descriptor page.

> **See also:** *Oracle9i Application Server mod_plsql User's Guide* which is part of the Oracle9*i* Application Server documentation library.

Choose one of the following options to resolve this issue:

- Access the database server and log on using the user name and password for the schema that owns the Oracle9*i*AS Portal page group. For example, if the schema name is "SCOTT", log on with the user name "SCOTT" and the appropriate password.
- Start SQL*Plus and execute the `inctxgrn.sql` script. This script is located in `ORACLE_HOME\portal\admin\plsql\wws`. Running this script creates the Oracle Text preferences required for Oracle9*i*AS Portal.
- If you do not have access to the database server, but you do have a copy of the `sbrimtlx` script, you can connect to the database using SQL*Plus as the schema owner and run the following commands:

```
set serveroutput on size 10000
begin
  wwv_context_util.grantCtxRole(user);
end;
@@sbrimtlx

Replace (user) with the Oracle9iAS Portal schema owner, for example,
portal.
```

See also: Chapter 7, "Configuring the Search Features in Oracle9iAS Portal".

Problem: The date is out of sync with the system clock

Cause

This error may occur when the sysdate value in the database that contains the Oracle9*i*AS Portal meta data repository is incorrect.

Solution

Check to make sure that the sysdate value in the database that contains the Oracle9*i*AS Portal meta data repository is incorrect. Refer to the Oracle9*i* database documentation for more information on how to do this.

Problem: OPCA install hangs at Oracle Ultra Search phase

Cause

This can happen when Oracle Ultra Search is already installed in the destination database and the value for DISPLAY is not set correctly.

During Oracle Ultra Search installation, a dialog will appear to ask the users if they wants to overwrite the existing Oracle Ultra Search packages

If the DISPLAY was not set correctly, the install will stop because the installer is waiting for the user's decision.

Solution

Ultra Search supports silent mode installation which can only be used if there is no existing installation of Oracle Ultra Search in the database. For interactive mode, the "DISPLAY" environment variable needs to be set correctly.

Note that the GUI will not appear when the destination database does not contain Oracle Ultra Search, in that case there is no need for the user to set DISPLAY.

A

Oracle9*i* Application Server Configuration Files

A.1 Overview

This appendix provides information about the configuration files and tables which can affect the connection to and the behavior of the Oracle9*i* Application Server and its components in the middle-tier as well as on other machines to which it is connecting.

Specific topics covered include:

- Oracle HTTP Server Configuration File (httpd.conf)
- Oracle Database Connection File (tnsnames.ora)
- Oracle9iAS Single Sign-On Server Configuration Table
- Oracle9iAS Single Sign-On Server's Partner Application Table
- Local HOSTS File

A.1.1 Oracle HTTP Server Configuration File (httpd.conf)

The Oracle HTTP Server configuration file, `httpd.conf`, contains configuration information for running the Oracle HTTP Server. The contents of this file includes information about listening ports, server names, virtual hosts, proxy configurations, and the like. Also, configuring Secure Sockets Layer (SSL) support by defining information such as certificates and other HTTPS configuration directives is done in this file.

`ORACLE_HOME/Apache/Apache/conf/httpd.conf`

See also: Section 2.9, "Enabling Secure Socket Layer (SSL)".

A.1.2 Oracle Database Connection File (tnsnames.ora)

The `tnsnames.ora` file defines the entries that can be used as connect strings in the DADs.

Also, the `tnsnames.ora` file in the Oracle Home location containing your Oracle9*i* Application Server must have a connect string entry pointing to the database where your Oracle Portal installation is located.

In the C shell, for example, type the following at a command line prompt:

```
setenv TNS_ADMIN path
```

`path` points to the `tnsnames.ora` file. This command differs depending on the shell used.

A.1.3 Web Cache Config files.

The following Oracle9*i*AS Web Cache configuration file can be found in the `ORACLE_HOME/webcache` directory:

- `webcache.xml`
- `internal.xml`
- `internal_admin.xml`

See also: *Oracle9iAS Web Cache Administration and Deployment Guide* for details on configuring these files.

A.1.4 Oracle9*i*AS Single Sign-On Server Configuration Table

The `WWSEC_ENABLER_CONFIG_INFO$` table is the configuration table for the Single Sign-on enabler stack.

Each Partner Application to the Oracle9*i*AS Single Sign-On Server has such a table for configuration information. As such, one such table exists in the Oracle9*i*AS Portal schema as well as the Oracle9*i*AS Single Sign-On Server schema, since the Oracle9*i*AS Single Sign-On Server application is a Partner Application as well. This table defines the login URL for the Oracle9*i*AS Single Sign-On Server which this Partner Application is associated with.

It is important to understand how the LSNR_TOKEN is used in the enabler configuration table in order to plan what entries are required depending on your configuration.

This table may have more than one entry. There is one entry for each way the application's server is addressed. Understanding this requires a review of the authentication sequence. For the purposes of this discussion, the main flows include:

- Initial request to the requested URL.
- Redirect to the Oracle9*i*AS Single Sign-On Server for authentication.
- Redirect to Oracle9*i*AS Portal's success URL (wwsec_app_priv.process_signon).
- Redirect back to the requested URL.

The Oracle9*i*AS Single Sign-On Server (SSO) partner enabler APIs read the WWSEC_ENABLER_CONFIG_INFO$ table for configuration information. Similarly, in the Oracle9*i*AS Single Sign-On Server, the Oracle9*i*AS Single Sign-On Server's private APIs read the WWSSO_PAPP_CONFIGURATION_INFO$ table. In the latter table, the URL that should be redirected to each Partner Application.

Since each Partner Application's success URL is stored in the Oracle9*i*AS Single Sign-On Server's Partner Application configuration table, to support multiple host names for the Partner Application, each distinct host name requires its own Partner Application entry on the Oracle9*i*AS Single Sign-On Server so that each one can specify a success URL that has the same hostname as the Partner Application so that the session cookie can be scoped appropriately. Furthermore, the domain to which cookies are scoped includes the server name (ServerName) and port, so server.domain.com:80 is treated as a different cookie domain from server.domain.com:8080.

Each entry in the enabler configuration table is then selected based on the host name and port that was used by the Partner Application.

For example, let's say that you wanted Oracle9*i*AS Portal to be accessible from http://www.xyz.com as well as http://www.abc.com. In this case, two Partner Applications must be registered in the Oracle9*i*AS Single Sign-On Server. One is defined for the www.xyz.com host and the other for the www.abc.com host. Each one specifies a success URL that is appropriate:

- http://www.xyz.com/pls/portal/portal.wwsec_app_priv.process_signon for the www.xyz.com partner

- `http://www.abc.com/pls/portal/portal.wwsec_app_priv.process_signon` for the www.abc.com application

Each of these Partner Application entries on the Oracle9*i*AS Single Sign-On Server would have a distinct site id, site token, and encryption key. Oracle9*i*AS Portal's enabler configuration table has one row for each Partner Application, for example:

```
LSNR_TOKEN    SITE_ID   LS_LOGIN_URL …
www.xyz.com   1321      https://www.login.com/pls/…
www.abc.com   1322      https://www.login.com/pls/…
```

See also: Oracle9iAS Single Sign-On Application Developer's Guide included in the Oracle9*i* Application Server documentation library.

A.1.5 Oracle9*i*AS Single Sign-On Server's Partner Application Table

The configuration table on the Oracle9*i*AS Single Sign-On Server's side is the Partner Application Table, `WWSSO_PAPP_CONFIGURATION_INFO$`. Maintenance of this table is typically done using the Oracle9*i*AS Single Sign-On Server application's user interface for Adding or Editing Partner Applications.

For an initial installation on a single database instance, running the OPCA in the SSO_PARTNERCONFIG mode populates both the Oracle9*i*AS Single Sign-On Server's partner configuration table as well as Oracle9*i*AS Portal's enabler configuration table.

A.1.6 Local HOSTS File

The HOSTS file on a network host defines mappings of IP names to IP addresses. Normally, the association of IP name to IP address is provided by a Domain Name Server (DNS). In some of the configurations described in Section 3, "Configuring Oracle9iAS Portal using OPCA", a host may need to be addressed in an internal network with a domain name that is not defined within the internal network. In these cases, the server's HOSTS file can provide the necessary name resolution.

A.1.7 Using Oracle Enterprise Manager

You can use Oracle Enterprise Manager for administering <your component name>. Oracle Enterprise Manager provides a Web-based tool that allows you to perform some of the management tasks described in this book.

See also: *Oracle9i Application Server Administrator's Guide* for more information about Oracle Enterprise Manager.

B

Oracle9*i*AS Portal Installation and Configuration Scripts

After installing Oracle9*i*AS Portal with the Oracle9*i* Application Server installation, several scripts are available for post-installation configuration. For example, you may want to configure a new Oracle9*i*AS Portal instance, or update an existing Oracle9*i*AS Portal instance.

Note: For information about the Oracle9*i*AS Portal import and export scripts, see the Oracle9*i*AS Portal Online Help topic: *Exporting and importing in Oracle Portal*.

For information about the Oracle9*i*AS Portal upgrade scripts, visit the Oracle Technology Network at:

http://otn.oracle.com

For purposes of configuring Oracle9*i*AS Portal, the following scripts are useful, and are described in this appendix. In Oracle9*i*AS Portal most Portal configuration is done by using the Oracle9*i*AS Portal Configuration Assistant (OPCA).

> **See also:** Section 3.1, "The Oracle9iAS Portal Configuration Assistant (OPCA)" about the Oracle9*i*AS Portal Configuration Assistant modes and parameters.

The specific scripts covered in this appendix include:

- Associating Oracle9iAS Portal with an Oracle9iAS Single Sign-On Server
- Oracle9iAS Web Cache configuration scripts

B.1 Associating Oracle9*i*AS Portal with an Oracle9*i*AS Single Sign-On Server

---

Note: Single Sign-On Server and Portal From Different Versions cannot interoperate

Due to the interdependency of the Oracle9*i*AS Single Sign-On Server and Oracle9*i*AS Portal with Oracle Internet Directory (OID) in Oracle9*i* Application Server Release 2 (9.0.2), you must not associate Oracle9*i*AS PortalRelease 9.0.2 with an Oracle9*i*AS Single Sign-On Server Server (Login Server) from Oracle9*i* Application Server Release 1 (1.0.2.2) or earlier. Similarly, you must not associate earlier releases of Oracle9*i* Application Server Portal with the current release of Oracle9*i*AS Single Sign-On Server.

To allow Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server instances to be associated together, they must both be upgraded to Oracle9*i* Application Server Release 2. The upgrade scripts will be made available in the first maintenance release of Oracle9*i* Application Server Release 2.

Oracle9*i*AS Portal is a partner application to the Oracle9*i*AS Single Sign-On Server. As such, it needs to be associated with an Oracle9*i*AS Single Sign-On Server for authentication services. When Oracle9*i*AS Portall is installed it is automatically associated with the Oracle9*i*AS Single Sign-On Server in the associated infrastructure installation.

What was formerly called the *ssodatan* script, in previous versions of Oracle9*i*AS Portal, has been obsoleted and replaced by running the Oracle9*i*AS Portal Configuration Assistant (OPCA) in the SSOPARTNERCONFIG mode. When you install Oracle9*i*AS Portal, the step previously done by *ssodatan*, is done

automatically. However, after installation, there may be various reasons for associating the Oracle9*i*AS Portal with a different Oracle9*i*AS Single Sign-On Server, or needing to re-run the association because of a change in the hostname, port or protocol of the Oracle9*i*AS Single Sign-On Server.

The function performed by the script previously called ssodatax, is now performed by the following mode of OPCA:

SSOPARTNERCONFIG mode usage Example:

```
ptlasst.csh -i typical -mode SSOPARTNERCONFIG -s portal -sp portal -c
myhost.domain.com:1521:mySID -sdad portal -o orasso -odad orasso -host
myApache.domain.com -port 7777 -silent -verbose -sso_c
myhost.domain.com:1521:mySID -sso_h myApache.domain.com -sso_p 7777 -pa orasso_
pa -pap orasso_pa -ps orasso_ps -pp orasso_ps -pd portal_dblink -p_tns orasso_ps
-s_tns portal -iasname myiASInstance.host
```

where

Table B–1 OPCA SSOPARTNERCONFIG mode parameters

Parameter	Description	Default value
`-i install_type`	Installation Type.	CUSTOM
`-s portal_schema`	Oracle Database schema for Oracle9*i*AS Portal database objects.	`portal`
`-sp portal_password`	Password for Oracle9*i*AS Portal schema.	`portal_schema`
`-c portal_connect`	**Mandatory** connect string to connect to the Oracle9*i*AS Portal repository.	`hostname:port:sid`
`-sdad portal_dad`	DAD for the Oracle9*i*AS Portal schema.	`portal_schema`
`-o sso_schema`	Oracle Database schema for Oracle9*i*AS Single Sign-On Server objects.	`orasso`
`-odad sso_dad`	DAD for Oracle9*i*AS Single Sign-On Server objects.	`sso_schema`

Table B–1 OPCA SSOPARTNERCONFIG mode parameters

Parameter	Description	Default value
`-host portal_host`	**Mandatory** hostname of the HTTP server for Oracle9*i*AS Portal.	`N/a`
`-port portal_port`	Port of the HTTP server for Oracle9*i*AS Portal.	`7777`
`-silent`	Run mode for the OPCA.	`N/a`
`-verbose`	Log mode for the OPCA.	`N/a`
`-sso_c sso_connect`	Connect string to connect to the Oracle9*i*AS Single Sign-On Server repository.	`N/a`
`-sso_h sso_host`	Hostname of the HTTP server for the Oracle9*i*AS Single Sign-On Server.	`N/a`
`-sso_p sso_port`	Port of the HTTP server for the Oracle9*i*AS Single Sign-On Server.	`N/a`
`-pa papp_schema`	SSO Partner Application Registration Schema	`sso_schema_PA`
`-pap papp_password`	Oracle9*i*AS Single Sign-On Server PA Schema Password	`papp_schema`
`-ps pstore_schema`	Password Store Schema.	`sso_schema_PS`
`-pp pstore_password`	Password Store Password.	`ptore_schema`
`-pd pstore_dblink`	Database link from Oracle9*i*AS Portal schema to Password Store.	`portal_schema_DBLINK`
`p_tns pstore_tns`	TNS connect string to Password Store.	`N/a`
`-s_tns portal_tns`	TNS connect string to Portal.	`N/a`

Whereas the old `ssodatax` required you to setup the partner application entry in the SSO server and then invoke the script with the site_id, site_token and encryption_key obtained from partner application registration, the

SSOPARTNERCONFIG mode of `ptlasst.csh`(OPCA) no longer requires partner application registration to be a two-step process.

The Oracle9*i*AS Single Sign-On Server now provides a schema ORASSO_PA (default) for accessing the partner application registration procedure. You will need to get the password to this schema and an appropriate connect string to the Oracle9*i*AS Single Sign-On Server instance to register the Oracle9*i*AS Portal entry.

Note that if the Oracle9*i*AS Single Sign-On Server hostname changes, you will also need to run `ssocfg.sh` on the Oracle9*i*AS Single Sign-On Server.

> **See also:** *Oracle9iAS Single Sign-On Administrator's Guide* for more details on running the `ssocfg.sh` script.

B.2 Oracle9*i*AS Web Cache configuration scripts

This section shows how instead of running OPCA in the MIDTIER mode to adjust Oracle9*i*AS Web Cache specific settings, such as the Oracle9*i*AS Web Cache host, or Oracle9*i*AS Web Cache invalidation port, you can choose to run Oracle9*i*AS Web Cache configuration scripts to configure Oracle9*i*AS Portal to work with Oracle9*i*AS Web Cache. Furthermore, it describes how you can disable Oracle9*i*AS Web Cache and how you can manage the invalidation message processing job by using the script `cachjsub.sql`:

Using cachseed.sql

Using cachset.sql

Managing the Invalidation Message Processing Job Using cachjsub.sql

B.2.1 Using cachseed.sql

With the `cachseed.sql` script you can modify all the Oracle9*i*AS Web Cache specific configuration parameters. `cachseed.sql` is located in the `ORACLE_HOME/portal/admin/plsql/wwc` directory. The script takes 6 arguments that are listed in the table below:

Table B–2 cachseed.sql arguments

argument	description
hostname	Oracle9*i*AS Web Cache hostname. For example *webdbsvr1.us.oracle.com*.

Table B–2 cachseed.sql arguments

argument	description
invalidationport	Oracle9*i*AS Web Cache invalidation port. For example *3002*.
adminport	Oracle9*i*AS Web Cache administration port. For example *3001*.
password	Oracle9*i*AS Web Cache invalidator password For example *invalidator.*
enable_wc_caching	Flag to turn Oracle9*i*AS Web Cache caching on or off in Oracle9*i*AS Portal. For example *off*
portal_dad	The dad name used to access the portal. For example *moc*

Example of running cachseed.sql

```
@cachseed.sql webdbsvr1.us.oracle.com 3002 3001 invalidator off moc
```

B.2.2 Using cachset.sql

The script `cachset.sql` is used to turn the use of Oracle9*i*AS Web Cache to on or off. The script can be found in the `ORACLE_HOME/portal/admin/plsql/wwc` directory.

To use `cachset.sql` connect to SQL*Plus as the schema owner and run `cachset.sql as follows:`

```
SQL>@cachset.sql
```

At the prompt enter *on* to enable the use of Oracle9*i*AS Web Cache and *off* to disable it.

B.2.3 Managing the Invalidation Message Processing Job Using cachjsub.sql

Oracle9*i*AS Portal uses caching to improve its performance. One type of caching used is the invalidation based caching. In this type of caching Oracle9*i*AS Portal Caches various objects (pages, portlets, etc) for a set amount of time. When these objects are requested they are retrieved from the Cache, if available, otherwise they are regenerated from the Oracle9*i*AS Portal repository. The Cache for these objects

will expire when the *maxcache* time has been reached, or when the objects are explicitly invalidated (expired) via invalidation messages.

Oracle9*i*AS Portal uses invalidation messages when it needs to expire objects in the Cache. invalidation messages are categorized as hard and soft invalidations. Hard invalidations take effect immediately, i.e. the objects which they intend to invalidate expire from Cache immediately. Soft invalidations take effect when they are processed by the invalidation processing job. The frequency by which the invalidation job executes is configurable. This is done via the `cachjsub.sql` script. Follow the following steps to change the execution frequency of the invalidation processing job:

1. Locate the following directory:

   `ORACLE_HOME/portal/admin/plsql/wwc`

2. On the database where the Portal schema is installed, log on to SQL*Plus with the appropriate user name and password for that schema. For example:

   ```
   sqlplus portal/portal
   ```

3. Enter the following command to update the execution frequency of the invalidation job:

   ```
   SQL> @cachjsub.sql <start_time> <start_time_fmt> <interval_mins>
   ```

   `cachjsub.sql` takes three parameters:

   - *start_time* is either when the first job should be run or 'START'.
   - *start_time_fmt* is the date format to be applied to the value of *start_time*.
   - *interval_mins* is how many minutes each run is scheduled apart.

   Note: If 'START' is provided for 1st parameter, the 2nd parameter is ignored and it will default the start time to the current time.

   Example1:

   ```
   SQL> @cachjsub.sql START null 120
   ```

   Example2:

   ```
   SQL> @cachjsub.sql '02-22-2003 7:30' 'MM-DD-YYYY HH:MI' 1440
   ```

B.3 Using oidprovtool to Create a Subscription Profile

Oracle9*i*AS Portal needs to subscribe to OID, in order to be aware of any changes in OID data.

One of the steps in setting this up is running a tool named `oidprovtool`. It will be located in:

```
ORACLE_HOME/bin
```

> **See also:** For the complete overview of what needs to be done to set up a subscription profile, refer to Section 2.7.1, "Setting up a Subscription Profile using oidprovtool".

The following table contains the valid parameter values for running the oidprovtool for creating the subscription profiles as required by Oracle9*i*AS Portal.

Table B–3 oidprovtool parameters

Parameter	Value	Example
operation	`create`	create
ldap_host	*`oid_host`*	portaloid
ldap_port	*`oid_port`*	389
ldap_user_dn	*`oid_admin_dn`*	cn=orcladmin
ldap_user_password	*`oid_admin_password`*	welcome1
application_dn	*`portal_application_dn`*	orclApplicationCommonName=PORTAL,cn=Portal,cn=Products,cn=OracleContext
organization_dn	*`subscriber_dn`*	"dc=mycompany,dc=com"
interface_name	*`portal_schema`*`.WWSEC_OID_SYNC`	PORTAL.WWSEC_OID_SYNC
interface_type	`PLSQL`	PLSQL
interface_connect_info	*`synchronization_interval_in_seconds`*	portaldbhost:1521:s901dev8:PORTAL:portalpassword
schedule	*`HOST`*`:`*`PORT`*`:`*`SID`*`:`*`USER_ID`*`:`*`PASSWORD`*	60

Table B–3 oidprovtool parameters

Parameter	Value	Example
event_subscription	"USER:*subscriber_dn*:DELETE"	"USER:dc=mycompany,dc=com:DELETE"
event_subscription	USER:*subscriber_dn*:DELETE	"GROUP:dc=mycompany,dc=com:DELETE"
event_subscription	USER:*subscriber_dn*:MODIFY(orclDefaultProfileGroup,userpassword)	"USER:dc=mycompany,dc=com:MODIFY(orclDefaultProfileGroup,userpassword)"
event_subscription	GROUP:*subscriber_dn*:MODIFY(uniqueMember)	"GROUP:dc=mycompany,dc=com:MODIFY(uniqueMember)"

Please note that the parameter values, or parts thereof, that are in italics, must be replaced by their value. All other parts, including quotation marks, must be entered as is in the call to oidprovtool. Thus using the above examples of values the complete call is as follows (please treat this as a single continuous line):

```
oidprovtool operation=create ldap_host=portaloid ldap_port=389 ldap_user_
dn=cn=orcladmin ldap_user_password=welcome1 application_
dn=orclApplicationCommonName=PORTAL,cn=Portal,cn=Products,cn=OracleContext
organization_dn="dc=mycompany,dc=com" interface_name=PORTAL.WWSEC_OID_SYNC
interface_type=PLSQL interface_connect_
info=portaldbhost:1521:s901dev8:PORTAL:portalpassword schedule=60 event_
subscription="USER:dc=mycompany,dc=com:DELETE" event_
subscription="GROUP:dc=mycompany,dc=com:DELETE" event_
subscription="USER:dc=mycompany,dc=com:MODIFY(orclDefaultProfileGroup,userpasswo
rd)" event_subscription="GROUP:dc=mycompany,dc=com:MODIFY(uniqueMember)"
```

> **See also:** For a complete list of all the possible options for oidprovtool, refer to the *Oracle Internet Directory Administrator's guide* in the Oracle9i Application Server documentation library.

B.4 Disabling the IP Check of Cookie Validation

As part of the process of validating the session cookie of a user's request (even if that user is PUBLIC) Portal performs a comparison between the IP address stored in the cookie with the IP address of the current client. Only if the two value are the same will Oracle9*i*AS Portal consider the request legitimate.

When a proxy exists between the user's client and the portal the IP address stored in the session cookie is that of the proxy, and not that of the client.

Some proxy systems make use of multiple servers each with different IP addresses. In these circumstances it is conceivable that the original request from a user's client (the request that causes the session cookie to be created) is routed through one proxy server and that a subsequent request is routed through another, separate, proxy server. In these cases, the IP addresses compared by Oracle9*i*AS Portal will differ and the request will raise a security violation during the IP checking step and access to the page will be denied.

Depending on the network configuration into which the Oracle9*i* Application Server is installed, it may be necessary to disable IP checking in cookie validation.

To change the state of IP checking in cookie validation, you need to use SQL*Plus to update data in both the portal schema and the SSO schema as detailed in the table below.

Table B–4 Enabling and Disabling the IP Check

	Portal Schema	SSO Schema

Enable IP Checking	`update wwsec_enabler_ config_info$`
	`set url_cookie_ip_check = 'Y';`
	`commit;`
		`update wwsec_enabler_ config_info$`
		`set url_cookie_ip_check ='Y';`
		`update wwsso_ls_ configuration_info$`
		`set cookie_ip_check = 'Y';`
		`commit;`

Disable IP Checking	`update wwsec_enabler_ config_info$`
	`set url_cookie_ip_check = 'N';`
	`commit;`
		`update wwsec_enabler_ config_info$`
		`set url_cookie_ip_check ='N';`
		`update wwsso_ls_ configuration_info$`
		`set cookie_ip_check = 'N';`
		`commit;`

B.5 Using the secupoid.sql script

By default, Oracle9*i*AS Portal connects to OID using LDAP without SSL. If the OID server is configured for an SSL port, though, Oracle9*i*AS Portal can be configured to use LDAP over SSL, also known as LDAPS.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for a detailed description on how to configure OID for an LDAPS port.

To configure Oracle9*i*AS Portal to use SSL to connect to OID, you must run the `secupoid.sql` script. This script allows you to change the following Oracle9*i*AS Portal configuration parameters related to OID:

- OID host name
- OID port
- application OID password
- SSL setting

When you install Oracle9*i*AS Portal, it is automatically the associated with an OID server. However, you may want to change some settings, such as whether to use SSL, after installation. To change to an SSL connection for OID, simply run the secupoid.sql script in the PORTAL schema to specify the LDAPS port instead of the LDAP port, and indicate that you want to use SSL.

B.5.1 Running the secupoid.sql script

The section that follows illustrates a sample execution of `secupoid.sql` from SQL*Plus.

In the example, OID was initially configured to run LDAP on port 389. Later, an LDAPS port was activated on 636. Since the server name does not change, we retain the old value, update the port, and indicate that we want to use SSL by setting the Use SSL? value to Y. When you run the script, it displays the current configuration and lets you replace any of the configurable settings. The script also allows you to update Oracle9*i*AS Portal's OID Cache after running it. Since activating SSL does not change any of the OID information cached by Oracle9*i*AS Portal, it is not usually necessary to refresh the Cache in this case.

```
SQL> @secupoid
Current Configuration
--------------------
OID Host: oid.domain.com
```

```
OID Port: 389
Application DN:
orclApplicationCommonName=PORTAL,cn=Portal,cn=Products,cn=OracleContext
Application Password: 3E8C2D1B87CB61011757239C5AA9B390
Use SSL? N

PL/SQL procedure successfully completed.

Updating OID Configuration Entries
Press [Enter] to retain the current value for each parameter
For SSL Connection to LDAP, specify "Y"es or "N"o
------------------------------------------------
Enter value for oid_host:
Enter value for oid_port: 636
Enter value for app_password:
Enter value for use_ssl_to_connect_to_ldap: Y
Enter value for refresh_with_new_settings: N

PL/SQL procedure successfully completed.

No errors.
```

After executing the script, Oracle9*i*AS Portal is configured for LDAPS access of OID.

See also:

- the *Oracle9i Application Server Security Guide* in the Oracle9*i* Application Server documentation library.
- Section 3.2.7, "SYSOBJECTS OPCA Mode"

B.6 Using the secjsdom.sql script

If you have your OID and Oracle9*i*AS Portal servers residing in different domains, you must explicitly set the JavaScript domain for Oracle9*i*AS Portal such that it can resolve user and group lists of values.

For example, suppose that your installation has Oracle9*i*AS Portal configured to use a different Oracle HTTP Server than DAS. In this situation, you need to have a common domain so that the values can be transferred from the list of values displayed by DAS to the page displayed by Oracle9*i*AS Portal.

To create a single domain in this case, do the following:

1. Login to SQL*Plus as PORTAL.

2. Run the following SQL script:

```
secjsdom.sql <domain_name>
```

Performing this procedure enables you to run OID lists of values from Oracle9*i*AS Portal in either Netscape or MicroSoft Internet Explorer. When using lists of values, a transit window is displayed in addition to the list of values itself. The transit window is required to pass values to Oracle9*i*AS Portal without forcing pages to reset their domain.

See also:

Oracle9i Application Server Security Guide in the Oracle9*i* Application Server documentation library.

B.7 Modifying the Scope of the Portal Session Cookie

In cases where you want access to the same portal from two middle-tiers at the same time, or if you want to open the portal cookie domain as required by the PL/SQL Adapter functionality you need to define the scope of the Oracle9*i*AS Portal session cookie to be sent to all the middle-tier servers involved in the architecture. By default, the session cookie is scoped to the host from which it was generated which is typically the root path.

For example, if the cookie was generated from `www.oracle.com`, then the cookie domain is `www.oracle.com`. However, let's say that another server, `portal.oracle.com` is also a middle-tier server that needs to get access to that session cookie, then the cookie domain would need to be widened so that the `portal.oracle.com` server can also see the cookie.

Follow these steps to modify the scope of the portal session cookie:

1. Locate the following directory:

```
ORACLE_HOME/portal/admin/plsql/wwc
```

2. On the database where your Oracle9*i*AS Single Sign-On Server schema is installed, log on to SQL*Plus with the appropriate username and password. For example:

```
sqlplus nodea/nodea
```

3. Enter the following command:

```
SQL> @ctxckupd
Oracle Portal
```

```
Current Settings for Portal Session Cookie:
Cookie Domain : Only send cookie back to originating host:port
Enter the domain for the session cookie: .oracle.com
Settings changed to
Cookie Domain : .oracle.com
SQL>
```

This allows you to set the cookie domain for the session cookie. In the example above, the cookie domain is set to `.oracle.com`.

Note: If you want to use different listeners or keep the session cookie throughout different domains, specify a Cookie Domain to be the host name only. For example, if you access Oracle9*i*AS Portal from two machines:

- `machine1.us.oracle.com:3000`
- `machine2.us.oracle.com:4000`

When running `ctxckupd.sql`, set the cookie domain to `.us.oracle.com`.

See also: Section 4.2.3, "Setting the Cookie Domain".

B.8 Managing the Session Cleanup Job

Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On Server perform session management similar to other web-based applications. Sessions are tracked using cookies. Session information is stored in a table in the Portal and Oracle9*i*AS Single Sign-On Server schema. When a user logs out, the session information is marked inactive. A DBMS job subsequently cleans up the inactive rows.

The session table accumulates a number of rows that are flagged as active. When a user shuts down the browser instead of logging out, the row is "active", even though it is not actually in use. The cleanup job cleans up the active rows that are older than a specified duration.

When Oracle9*i*AS Portal is installed, a DBMS job is installed to perform session cleanup of the session table, `WWCTX_SSO_SESSION$`. The cleanup job is set to run every 24 hours. The first scheduled cleanup occurs 24 hours after the installation of the job.

When the job runs, it deletes all inactive sessions, and all sessions marked active (`WWCTX_SSO_SESSION$.ACTIVE = 1`), that are older than 7 days (`WWCTX_SSO_SESSION$.SESSION_START_TIME < sysdate - 7`).

These default settings can be modified by running some job management scripts in the Portal schema to manage Portal sessions, or in the Oracle9*i*AS Single Sign-On Server schema to manage Oracle9*i*AS Single Sign-On Server sessions. They utilize the same session management infrastructure.

Follow these steps to obtain the current cleanup job information:

1. Locate the following directory:

   ```
   ORACLE_HOME/portal/admin/plsql/wwc
   ```

2. On the database where the Portal or Oracle9*i*AS Single Sign-On Server schema is installed, log on to SQL*Plus with the appropriate user name and password for that schema. For example:

   ```
   sqlplus portal/portal
   ```

3. Enter the following command to get the current job information:

   ```
   SQL> @ctxjget
   The session cleanup job is job ID 7381
   dbms_job.isubmit(job=>7381,what=>'begin execute immediate''begin
   wwctx_sso.cleanup_sessions(p_hours_old => 168); end;''; exception when
   others then null; end;',next_date=>to_date('2001-04-17:14:07:20',
   'YYYY-MM-DD:HH24:MI:SS'),interval=>'SYSDATE + 24/24',no_parse=>TRUE);

   PL/SQL procedure successfully completed.
   ```

The command results in the display of the currently installed job information, as returned by the DBMS_JOB package. It indicates which procedure is executed, what parameters are passed to it, and when the next invocation is to occur. This particular example indicates that the job is to cleanup active sessions which are a week old (168 hours). It also indicates that the next scheduled job execution is on 4/17/2001 at 5:14 pm, and the job should run every 24 hours thereafter.

If the job execution needs to be modified, either to adjust the age of sessions that should be deleted, or to increase or decrease the frequency of cleanup, you can run the `ctxjsub.sql` script to submit modified execution parameters.

Follow these steps to submit modified job execution parameters:

1. Locate the following directory:

```
ORACLE_HOME/portal/admin/plsql/wwc
```

2. On the database where the Portal or Oracle9*i*AS Single Sign-On Server schema is installed, log on to SQL*Plus with the appropriate user name and password for that schema. For example:

   ```
   sqlplus portal/portal
   ```

3. Enter the following command to submit new cleanup job information:

   ```
   @ctxjsub <hours_old> <start_time> <time_format> <interval_hours>
   ```

Table B–5 lists the `ctxjsub` parameters.

Table B–5 ctxjsub parameters

Parameter	Description
`hours_old`	The age of an active session that should be deleted.
`start_time`	The time that the next job should run.
`time_format`	The time format string that specifies how `start_time` is formatted.
`interval_hours`	The amount of time, in hours, between runs of the cleanup job.

For example:

```
SQL> @ctxjsub 200 '04/17/2001 10:00' 'MM/DD/YYYY HH24:MI' 12
Created path for job id.
DBMS_JOB id = 7381
Cleanup job updated. Job ID = 7381

PL/SQL procedure successfully completed.
```

The cleanup job submission script can be run any number of times to modify the execution parameters. Each invocation updates the job information associated with the job ID for the cleanup job. This job ID is maintained in the preference store so that the job information is updated instead of submitting multiple jobs.

You can also specify a `start_time` of 'START', in which case, the `time_format` parameter is ignored, but you still need to pass it a value (such as 'NOW'). The result is to run the job `<interval_hours>` hours from now:

```
SQL> @ctxjsub 168 START NOW 24
```

This submits the job as it does in the installation.

If you want the cleanup job to execute immediately, then obtain the job ID by calling `ctxjget.sql`. Once you know the job ID, you can execute the job by issuing the following command in the product schema:

```
SQL> exec dbms_job.run(7381);
```

In the preceding example, 7381 is the job ID returned by the call to `ctxjget.sql`. When you execute a job in this manner, the next automated invocation of the job occurs at `interval_hours` after this manual invocation. To run the job on the original schedule, you need to resubmit the `start_time` desired using `ctxjsub.sql`.

B.9 Timing and Caching Statistics

All Oracle9*i*AS Portal pages can be run in a special mode in which timing and caching information is displayed. If you want to see this debug information on every page you can set the Parallel Page Engine Parameter showPageDebug to true in the web.xml file.

> **See also:** Section 6.2.1.1, "Setting PPE Configuration Parameters" for information on how to configure the PPE settings.

If you just want to see the debug information for a few select pages and portlets, you can run a page with this debugging information enabled by adding "`&_debug=0`", or "`&_debug=1`" to the end of the Oracle9*i*AS Portal page url. For example if you want to see the timing statistics for the page `http://abc.com/servlet/page?_pageid=21`, you would append an ampersand (&) and "`_debug=0`", or `"_debug=1"` to the page url like this:

```
http://abc.com/servlet/page?_pageid=21&_debug=0
```

or:

```
http://abc.com/servlet/page?_pageid=21&_debug=1
```

The difference between "&_debug=0" and "&_debug=1" is that with the first option, the *debug=0* parameter is not passed to either the portlets or the page meta data. The Cache statistics are therefore reflected in the most accurate way. In the second option, "&_debug=1", the parameter *debug=1* is passed through to the portlet itself, which can affect the caching information. Since the parameter is passed to the portlets, it can be used by the portlet developers to display portlet-specific debugging information. For instance if you had a portlet, and you wanted to know how long a particular query takes, or if you wanted to debug a specific issue, you

could write some debugging code to print data to the browser when the debug=1 parameter is passed in.

The following statistics are available when the portal page is run in debug mode:

- Portlet Statistics
- Page Statistics
- Summary Statistics

The following image shows a page that is running in the "_debug=0" mode:

Figure B–1 Portal Page Running in Debug Mode

B.9.1 Portlet Statistics

In the above example you can see a number of Portlet related statistics listed under each portlet. Each Portlet has a unique internal reference identification number. This number is used in the "Information for Portlet" summary. For the portlet in the top left corner of the above image you can see that this number is 6256.

For each portlet the following statistics are listed:

B.9.1.1 Portlet Timing information

- **Portlet Timing** (msecs) (wait msecs)

  Indicates how many milli-seconds to retrieve the portlet, and how long the request was queued, also in milli-seconds.

- **Timing Status**

  This is deprecated and no longer in use.

- **XSLT Timing** (msecs)

  Displays the number of milli-seconds needed to retrieve the XSL stylesheet, in case the portlet is an XML portlet.

B.9.1.2 Portlet Caching information

- **Portlet Cache status** Web Cache (values) File System Cache (values)

  This is the Cache status from both Oracle9*i*AS Web Cache and the modplsql file Cache.

  Valid values for Oracle9*i*AS Web Cache are:

  - MISS, or NEW [M] indicating a Cache miss in Oracle9*i*AS Web Cache and that the content that is generated by the portlet is new.
  - MISS, or STALE [G] indicating a Cache miss, due to stale content in Oracle9*i*AS Web Cache.
  - HIT [H] indicating a Oracle9*i*AS Web Cache hit.

  Valid values for File System Cache are:

  - HIT_PING indicating a Cache hit for a validation based portlet.
  - HIT_EXPIRES indicating a Cache hit for an expiry based portlet.
  - MISS_STALE indicating a Cache miss due to stale content in the Cache. This applies to both expiry as well as validation based portlets.

- MISS_NEW indicating a Cache miss and that the content that is generated by the portlet is new. This applies to both expiry as well as validation based portlets.

If a portlet uses the File System Cache, then the above information will be listed. Otherwise it will be null.

If there is a hit on Oracle9*i*AS Web Cache, no details about File System Cache will be displayed as the content is served directly out of Oracle9*i*AS Web Cache. Additionally, if a portlet does not use Oracle9*i*AS Web Cache, then no Web Cache information will be printed.

- **From Cache:Web Cache** Cache Expires (secs), Age in Cache (secs) File System Cache (values)

  Information from both Oracle9*i*AS Web Cache and File System Cache will be printed here based on the type of caching that the portlet uses.

  > **See also:** *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i* Application Server documentation library.

  "Cache Expires" lists the number of seconds after which the portlet content in Oracle9*i*AS Web Cache will expire.

  "Age in Cache" lists the number of seconds that the portlet content has been Cached in Oracle9*i*AS Web Cache.

  "File System Cache" displays the information obtained from the File System Cache about Cache key, Cache expiry and about the Cache level in case of a Cache hit, with the Cache Status of either HIT_PING, or HIT_EXPIRES.

  In case of a Cache hit, the Cache key and Cache level (for Validation based portlets) and Cache Expires and Cache Level (for expiry based portlets) are displayed, with the Cache Status value of either HIT_PING or HIT_EXPIRES.

  For Validation and Expires based portlets, "None" is printed, when there is a Cache miss due to the portlet content being new. (Cache Status: MISS_NEW) The portlet is contacted to get the new Cache key, Cache expiry and Cache level.

  For Validation based portlets, if the content in the Cache has become stale resulting in a Cache miss, the current values in the Cache for Cache key and Cache level are displayed. In this case the portlet is contacted to get the updated Cache key and the level (Cache Status: MISS_STALE).

  For Expires based portlets, when the content in the Cache has become stale resulting in a Cache miss, a value of "INVALID" in the expires field and Cache

level are displayed. In this case the portlet is contacted to get the updated Cache Expiry and Cache level (Cache Status: MISS_STALE).

- **From Portlet**: (Cache Key) (Cache Level)

  This is the information obtained from the portlet about File System Cache Key Cache Expiry and Cache Level when there is a Cache miss and when portlet is contacted for the updated, or new values (Cache Status: MISS_NEW, or MISS_STALE). Note that there is no Oracle9*i*AS Web Cache related information displayed in this section.

  For Validation based portlets, when there is a Cache hit and if the ping is successful, meaning the content in the Cache is still valid, then the portlet does not return a new Cache key and level, instead it will indicate that the Cache is still valid. In this case, "Ping Success" is displayed (Cache Status: HIT_PING).

  For Expires based portlets, when there is a Cache hit and if the content has not expired, then the portlet is not contacted for the content. In this case, "Not contacted" is displayed (Cache Status: HIT_EXPIRES).

  Following are a few examples that show different Caching scenarios and the resulting output. Note that the other Page and Portlet related output is not shown here.

  Example Caching Information Debug Output 1

  - **Portlet Cache:** File System Cache, **Caching Type:** Validation based, **Status:** MISS, STALE.

    ```
    Caching information for portlet:
    Portlet Cache status: File System Cache:- MISS,STALE
    From Cache: File System Cache:- Cache Key: 42, Cache Level: USER
    From Portlet: Cache Key: 44, Cache Level: USER
    ```

  Example Caching Information Debug Output 2

  - **Portlet Cache:** File System Cache, **Caching Type:** Expires based, **Status:** MISS, NEW.

    ```
    Caching information for portlet:
    Portlet Cache status:File System Cache:-  MISS,NEW
    From Cache: File System Cache:-None
    From Portlet: Cache Expires: 1, Cache Level: USER
    ```

  Example Caching Information Debug Output 3

- **Portlet Cache:** File System Cache, Web Cache, **Caching Type:** Validation and Invalidation based, **Status:** MISS, NEW in File System Cache and Web Cache.

```
Caching information for portlet:
Portlet Cache status: Web Cache:- MISS,NEW [M], File System Cache:-
MISS,NEW
From Cache: Web Cache:- Cache Expires: 86400 secs, Age in Cache: 0 secs
, File System Cache:- None
From Portlet: Cache Key: 9.0.2.2.1502:04:18:09:19:56, Cache Level:
SYSTEM
```

Example Caching Information Debug Output 4

- **Portlet Cache:** Web Cache, **Caching Type:** Invalidation based, **Status:** HIT in Web Cache.

```
Caching information for portlet:
Portlet Cache status: Web Cache:- HIT [H]
From Cache: Web Cache:- Cache Expires: 86400 secs, Age in Cache: 58 secs
From Portlet: -
```

B.9.2 Page Statistics

Every page has a unique internal reference identification number, similar to the portlets on the page, shown in the image above.

For the page the following statistics are listed:

- **Elapsed Time** (msecs)

  This is the total amount of time required to generate the page calculated in the Parallel Page Engine (PPE). The actual generation time in the browser can be higher, due to network overhead.

  Elapsed time is made up of Page meta WAIT time and Stream time. Page meta WAIT time is the time taken to wait on content via an HTTP connection. Stream time is the time taken streaming and assembling the content pieces. Stream time is in turn composed of the following elements:

  - Page meta time
  - Time waiting for portlets to complete
  - Time taken streaming content to the browser

Effectively, elapsed time is the total amount of time (in milli-seconds), that it takes to put the page together, from the time the request was received to the last byte being written to the browser.

- **Page meta-time** (msecs) (wait = msecs)

  Displays the time that it takes to retrieve the page meta data. The wait time (msecs) represents how long the request was queued.

- **Page meta Cache Status** (Web Cache values), (Cache Expires msecs), (Age in Cache msecs) , (File System Cache values)

  Represents the Cache status from both Oracle9*i*AS Web Cache and modplsql file Cache. Valid values for Oracle9*i*AS Web Cache are MISS, or NEW and HIT. Valid values for file Cache are HIT, or PING, and MISS, or STALE. The Web Cache Expires value and the Age in Cache are both measured in milli-seconds.

- **Login meta-time** (msecs) (wait msecs)

  Displays the time (in milli-seconds) that it takes to retrieve the login meta data. The wait time represents the total amount of time (in milli-seconds) that the request spend in the request queue.

- **Login meta Cache Status**

  Similar to Page meta Cache status above, represents the Cache status for the login meta data from both Web Cache and modplsql file Cache.

B.9.3 Additional Summary Statistics

- **Stream info** (msecs)

  Represents (in milli-seconds) how long it takes for the page to stream to the browser.

- **processing** (msecs)

  Processing time (in milli-seconds) for streaming.

- **write** (msecs)

  The write lines can repeat several times. The lines represent each physical buffer write to the stream itself. This are one set for each buffer write.

- **flush** (msecs)

  The flush logs indicate that the writing stream was flushed. this is logged to keep track of the number of network round trips.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W