Oracle9*i* Application Server Oracle

Using the PL/SQL Gateway

Release 1 (v1.0.2.2)

May 2001

Part No. A90099-01

ORACLE®

Using the PL/SQL Gateway, Release 1 (v1.0.2.2)

Part No. A90099-01

Primary Author: Cheryl Smith

Contributing Authors: Ron Decker, Pushkar Kapasi, Sanjay Khanna, Eric Lee, Kannan Muthukkaruppan

Contents

2 Securing Applications through the PL/SQL Gateway

3 Caching

4 Configuring the PL/SQL Gateway

5 Installing the PL/SQL Gateway

Send Us Your Comments

Using the PL/SQL Gateway, Release 1 (v1.0.2.2)

Part No. A90099-01

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 have any suggestions for improvement, indicate the document title and part number, and the chapter, section, and page number (if available). Send comments to:

- iasdocs_us@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 manual describes how to install, configure, and maintain the PL/SQL Gateway for Oracle9*i* Application Server 1.0.2.2. It contains the following chapters:

Chapter 1 - Provides an overview of the PL/SQL Gateway and its features.

Chapter 2 - Describes how to secure applications.

Chapter 3 - Specifies the different methods of caching in the PL/SQL Gateway.

Chapter 4 - Describes global PL/SQL Gateway settings, and individual Database Access Descriptor and Cache settings.

Chapter 5 - Explains how to install the PL/SQL Gateway.

Chapter 6 - Provides step-by-step instructions for creating and invoking a simple application that displays the contents of a database table in an HTML page.

Appendix A - Explains how to tune the performance of the PL/SQL Gateway.

Appendix B - Describes common problems and solutions.

Related Oracle Documents

For more information, see the following manuals:

Part Number	Title
A90101-01	*Oracle9i* Application Server, *PL/SQL Web Toolkit Reference*
A90215-01	*Oracle9i Application Server Installation Guide for Sun Sparc Solaris*
A90216-01	*Oracle9i Application Server Installation Guide for Windows NT*

Part Number	Title
A83709-05	*Oracle9i Application Server - Migrating from Oracle Application Server*
A87353-01	*Oracle9i Application Server - Overview*

Oracle Services and Support

Information about Oracle products and global services is available from:

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

The sections below provide URLs for selected services.

Oracle Support Services

Technical Support contact information worldwide is listed at:

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

Templates are provided to help you prepare information about your problem before you call. You will also need your CSI number (if applicable) or complete contact details, including any special project information.

Product and Documentation

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

- `http://store.oracle.com`

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

Product documentation can be found at:

- `http://docs.oracle.com`

Customer Service

Global Customer Service contacts are listed at:

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

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://technet.oracle.com`

OTN delivers technical papers, code samples, product documentation, self-service developer support, and Oracle key developer products to enable rapid development and deployment of application built on Oracle technology.

New Features in Oracle9*i* Application Server, Release 1.0.2.2

The 1.0.2.2 version of mod_plsql has the following new features:

Table 0–1 New Features

New Features	Refer to:
HTTP HEAD requests.	"POST, GET and HEAD Methods" on page 1-7.
New Stateless (preserve package state) mode of execution.	"Stateless (Preserve Package)" on page 1-4.
Per DAD level NLS_LANG, NLS_ TERRITORY and CHARSET.	"DAD Settings Accessible through the Configuration File" on page 4-8.
Host:Port:SID style connect string.	"Accessing Database Access Descriptor settings" on page 4-4.
Critical warnings/errors are now logged to Apache error logs. Support for Apache ErrorDocument directive.	"DAD Settings Accessible through the Configuration File" on page 4-8 and "Logging" on page B-3 for more information.
OWA package documentation	PL/SQL Web Toolkit Reference guide.

Conventions

The following conventions are used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Convention	Meaning
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

PL/SQL Gateway Overview

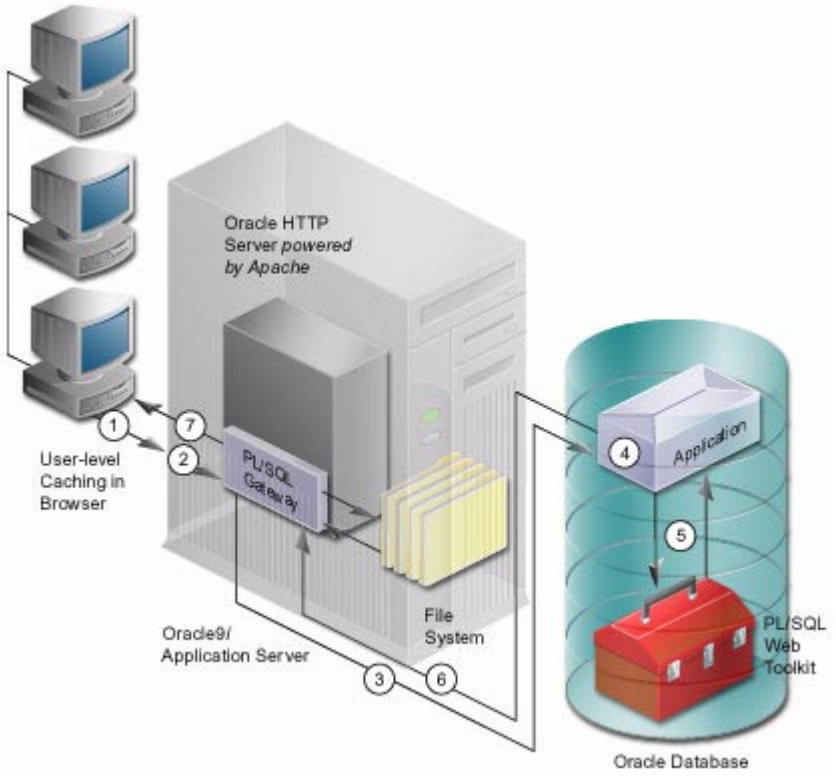
Oracle9*i* Application Server consolidates Oracle's middle-tier products into a single solution for the deployment of Web applications. The PL/SQL Gateway provides support for building PL/SQL-based applications on the Web. PL/SQL stored procedures can retrieve data from a database and generate HTTP responses containing data and code to display in a Web browser. The PL/SQL Gateway also supports other Oracle products such as Oracle Portal.

1.1 Processing Client Requests

The PL/SQL Gateway has two versions. The first involves mod_plsql, which is an Apache plug-in used to communicate with the database. It maps browser requests into database stored procedure calls over a SQL*Net connection. It is generally indicated by a */pls* virtual path.

The second version involves the use of mod_ose. There is an embedded PL/SQL Gateway that is shipped with mod_ose.

The following scenario provides an overview of what steps occur when a server receives a client request:



1. The Oracle HTTP Server receives a PL/SQL Server Page request from a client browser.
2. The Oracle HTTP Server routes the request to the PL/SQL Gateway.
3. The request is forwarded by the PL/SQL Gateway to the Oracle8i Database. By using the configuration information stored in your DAD, the PL/SQL Gateway connects to the database.
4. The PL/SQL Gateway prepares the call parameters, and invokes the PL/SQL procedure in the application. Refer to "Configuring the PL/SQL Gateway" on page 4-1 for more information.
5. The PL/SQL procedure generates an HTML page using data and the PL/SQL Web Toolkit accessed from the database.

6. The response is returned to the PL/SQL Gateway.
7. The Oracle HTTP Server sends the response to the client browser.

The procedure that the PL/SQL Gateway invokes returns the HTTP response to the client. To simplify this task, the PL/SQL Gateway includes the PL/SQL Web Toolkit, a set of packages (also called the owa packages) that you can use in your stored procedure to get information about the request, construct HTML tags, and return header information to the client. Install the toolkit in a common schema so that all users can access it.

1.2 PL/SQL Gateway Configurations

The use of mod_plsql or mod_ose determines which mode (stateful or stateless) is appropriate for you.

Table 1–1 Stateful and Stateless Modes

mod_plsql	mod_ose
Stateless (Reset Package - used for Oracle Portal) ■ implicit commit after request ■ clears all variables and cursors	Stateful **Note:** mod_ose can use stateless modes, but performance will suffer. ■ no implicit commit ■ clears OWA packages (not application specific ones)
Stateless (Preserve Package - Applications) ■ implicit commit after request ■ clears OWA packages (not application specific ones)	

1.2.1 Stateless (Reset Package)

In this mode, mod_plsql calls dbms_session.reset_package after each request to clear all OWA packages and any application-specific variables and cursors. Furthermore, an implicit commit is done if there are no errors in processing the request.

PL/SQL applications are not responsible for cleaning up after the request is done. With each request, they are starting in a clean session which the mod_plsql is responsible for. This mode is used by Oracle Portal 3.0.

1.2.2 Stateless (Preserve Package)

In this state, the mod_plsql calls htp.init after each request to only clear OWA packages, and not any application-specific variables and cursors. Furthermore, an implicit commit is done if there are no errors in processing the request.

PL/SQL applications are responsible for cleaning out the session at the end of request, but are not responsible for issuing commit in their code.

1.2.3 Stateful

In this state, mod_plsql calls htp.init after each request to only clear out OWA packages, and not any application-specific variables and cursors. However, there is no implicit commit done at the end of the request.

PL/SQL applications are responsible for cleaning out the session at the end of request. They are also responsible for issuing commit in their code. They cannot assume that they start in a clean session. They start in whatever state the last session left off.

This mode is used in the embedded PL/SQL Gateway that ships with mod_ose. When using mod_ose, the stateful mode is preferable because a new database session does not have to be created and destroyed for every HTTP request. For more information, see the Oracle8*i* Oracle Servlet Engine User's Guide in the Oracle 9*i* Application Server Documentation Library.

1.3 Database Access Descriptors

Each PL/SQL Gateway request is associated with a Database Access Descriptor (DAD), a set of configuration values used for database access. A DAD specifies information such as:

- the database alias (Net8 service name).
- a connect string if the database is remote.
- a procedure for uploading and downloading documents.

You can also specify a username and password information in a DAD. If they are not specified, the user is prompted to enter a username and password when the URL is invoked. For more information, refer to "Authenticating Users" on page 2-1.

1.4 Invoking the PL/SQL Gateway

To invoke the PL/SQL Gateway in a Web browser, input the URL in the following format:

protocol`://`*hostname*`[:`*port*`]/`*prefix*`/`*DAD*`/[[!][`*schema*`.][`*package*`.]`*proc_name*`[?`*query_string*`]]`

Table 1–2 Invoking the PL/SQL Gateway Parameters

Parameter	Description
protocol	Either `http` or `https`. For SSL, use https.
hostname	The machine where the Web server is running.
port (optional)	The port at which the application server is listening. If omitted, port 80 is assumed.
prefix	A virtual path to handle PL/SQL requests that you have configured in the Web server. `pls` is the default setting for this parameter. For example, you can configure the Web server to set `pls` as the prefix so that all requests containing the `pls` prefix are routed to the PL/SQL Gateway.
DAD	The DAD entry to be used for this URL.
! character (optional)	Indicates to use the flexible parameter passing scheme. See"Flexible Parameter Passing" on page 1-9 for more information.
schema (optional)	The database schema name. If omitted, name resolution for *package.proc_name* occurs based on the database user that the URL request is processed as.
package (optional)	The package that contains the PL/SQL stored procedure. If omitted, the procedure is stand-alone.
proc_name	The PL/SQL stored procedure to run. This must be a procedure and not a function. It can accept only IN arguments.

Table 1–2 Invoking the PL/SQL Gateway Parameters

Parameter	Description
?query_string (optional)	The parameters for the stored procedure. The string follows the format of the GET method. For example: ■ Multiple parameters are separated with the & character. Space characters in the values to be passed in are replaced with the + character. ■ If you use HTML forms to generate the string (as opposed to generating the string yourself), the formatting is done automatically. ■ The HTTP request may also choose the HTTP POST method to post data to the PL/SQL Gateway. See "POST, GET and HEAD Methods" on page 1-7 for more information.

Example 1: A Web server is configured with `pls` as a prefix and the browser sends the following URL:

```
http://www.acme.com:9000/pls/mydad/mypackage.myproc
```

The Web server running on `www.acme.com` and listening at port `9000` handles the request. When the Web server receives the request, it passes the request to the PL/SQL Gateway. This is because the `pls` prefix indicates that the Web server is configured to invoke the PL/SQL Gateway. The PL/SQL Gateway then uses the DAD associated with `mydad` and runs the `myproc` procedure stored in `mypackage`.

Example 2: Specify a URL without a DAD, schema, or stored procedure name.

```
http://www.acme.com:9000/pls/mydad
```

Then the default home page for the `mydad` DAD (as specified on the Gateway Configuration pages) displays.

Example 3: Specify a URL to invoke the default DAD's default home page:

```
http://www.acme.com:9000/pls
```

Generally, it does not matter what order the PL/SQL parameters are entered in the URL or the HTTP header since the parameters are passed by name. However, there are some exceptions to this rule. Refer to "Parameter passing" on page 1-7 for more information.

1.4.1 POST, GET and HEAD Methods

The POST, GET and HEAD methods in the HTTP protocol instruct browsers on how to pass parameter data (usually in the form of name-value pairs) to applications. The parameter data is generated by HTML forms.

PL/SQL Gateway applications can use any of the methods. Each method is as secure as the underlying transport protocol (http or https).

- When using the POST method, parameters are passed in the request body. Generally, if you are passing large amounts of parameter data to the server, use the POST method.
- When using the GET method, parameters are passed using a query string. The limitation of this method is that the length of the value in a name-value pair cannot exceed the maximum length for the value of an environment variable, as imposed by the underlying operating system. In addition, operating systems have a limit on how many environment variables you can define.
- When using the HEAD method, it has the same functionality as the GET method. The only difference is that only the HTTP status line and the HTTP headers are passed back. No content data is streamed back to the browser. This is useful for monitoring tools in which you are only interested if the request is processed correctly.

1.5 Transaction Mode

After processing a URL request for a procedure invocation, the PL/SQL Gateway performs a rollback if there were any errors. Otherwise, the Gateway performs a commit. This mechanism does not allow a transaction to span across multiple HTTP requests. In this stateless model, applications typically maintain state using HTTP cookies or database tables. For more information about stateful and stateless modes, see "PL/SQL Gateway Configurations" on page 1-3.

1.6 Parameter passing

The PL/SQL Gateway supports:

- **Parameter passing by name**

  Each parameter in a URL that invokes procedure or functions identified by a unique name. Overloaded parameters are supported. See "Parameter Passing by Name (Overloaded parameters)" on page 1-8 for more information.

- **Flexible parameter passing**

  Procedures are prefixed by a ! character. See "Flexible Parameter Passing" on page 1-9 for more information.

- **Large (up to 32K) parameters passing**

  See "Large Parameter Passing" on page 1-11 for more information.

1.6.1 Parameter Passing by Name (Overloaded parameters)

Overloading allows multiple subprograms (procedures or functions) to have the same name, but differ in the number, order, or the datatype family of the parameters. When you call an overloaded subprogram, the PL/SQL compiler determines which subprogram to call based on the data types passed.

PL/SQL allows you to overload local or packaged subprograms. Stand-alone subprograms cannot be overloaded. See the *PL/SQL User's Guide* in the Oracle Server documentation for more information on PL/SQL overloading.

You must give parameters different names for overloaded subprograms that have the same number of parameters. Because HTML data is not associated with datatypes, the PL/SQL Gateway does not know which version of the subprogram to call.

For example, although PL/SQL allows you to define two procedures using the same parameter names for the procedures, an error occurs if you use this with the PL/SQL Gateway.

```
-- legal PL/SQL, but not for the PL/SQL Gateway
CREATE PACKAGE my_pkg AS
  PROCEDURE my_proc (val IN VARCHAR2);
  PROCEDURE my_proc (val IN NUMBER);
END my_pkg;
```

To avoid the error, name the parameters differently. For example:

```
-- legal PL/SQL and also works for the PL/SQL Gateway
CREATE PACKAGE my_pkg AS
  PROCEDURE my_proc (valvc2 IN VARCHAR2);
  PROCEDURE my_proc (valnum IN NUMBER);
END my_pkg;
```

The URL to invoke the first version of the procedure looks similar to:

```
http://www.acme.com/pls/myDAD/my_pkg.my_proc?valvc2=input
```

The URL to invoke the second version of the procedure looks similar to:

```
http://www.acme.com/pls/myDAD/my_pkg.my_proc?valnum=34
```

1.6.1.1 Overloading and PL/SQL Arrays

If you have overloaded PL/SQL procedures where the parameter names are identical, but the data type is *owa_util.ident_arr* (a table of varchar2*)* for one procedure and a scalar type for another procedure, the PL/SQL Gateway can still distinguish between the two procedures. For example, if you have the following procedures:

```
CREATE PACKAGE my_pkg AS
  PROCEDURE my_proc (val IN VARCHAR2); -- scalar data type
  PROCEDURE my_proc (val IN owa_util.ident_arr); -- array data type
END my_pkg;
```

Each of these procedures has a single parameter of the same name, `val`.

When the PL/SQL Gateway gets a request that has only one value for the val parameter, it invokes the procedure with the scalar data type.

Example 1: Send the following URL to execute the scalar version of the procedure:

```
http://www.acme.com/pls/myDAD/my_proc?val=john
```

When the PL/SQL Gateway gets a request with more than one value for the val parameter, it then invokes the procedure with the array data type.

Example 2: Send the following URL to execute the array version of the procedure:

```
http://www.acme.com/pls/myDAD/my_proc?val=john&val=sally
```

To ensure that the array version executes, use hidden form elements on your HTML page to send dummy values that are checked and discarded in your procedure.

1.6.2 Flexible Parameter Passing

The PL/SQL Gateway supports flexible parameter passing to handle HTML forms where users can select any number of elements. To use flexible parameter passing for a URL-based procedure invocation, prefix the procedure with an exclamation mark (!) in the URL. You can use two or four parameters. The two parameter

interface provides improved performance with the PL/SQL Gateway. The four parameter interface is supported for compatibility.

Note: For questions about backwards compatibility with OAS, refer to "Using Flexible Parameters and the Exclamation Mark" on page B-2.

1.6.2.1 Two parameter interface

```
procedure [proc_name] is
      name_array  IN  [array_type],
      value_array IN  [array_type],
```

Table 1–3 Two Parameter Interface Parameters

Parameter	Description
proc_name (required)	The name of the PL/SQL procedure that you are invoking.
name_array	The names from the query string (indexed from 1) in the order submitted.
value_array	The values from the query string (indexed from 1) in the order submitted.
array_type (required)	The values from the query string (indexed from 1) in the order submitted.

Example: If you send the following URL:

```
http://www.acme.com/pls/myDAD/!scott.my_proc?x=john&y=10&z=doe
```

The exclamation mark prefix (!) instructs the PL/SQL Gateway to use flexible parameter passing. It invokes procedure *scott.myproc* and passes it the following two arguments:

```
name_array ==> ('x', 'y', 'z')
values_array ==> ('john', '10', 'doe')
```

1.6.2.2 Four parameter interface

The four parameter interface is supported for compatibility. If you are experiencing overhead problems due to using the four parameter interface, refer to "The Flexible Parameter Passing (four parameter) Overhead" on page B-10.

```
procedure [proc_name] is
         (num_entires IN NUMBER,
          name_array  IN  [array_type],
          value_array IN  [array_type],
          reserved in [array_type]);
```

Table 1–4 Four Parameter Interface Parameters

Parameter	Description
proc_name (required)	The name of the PL/SQL procedure that you are invoking.
num_entries	The number of name_value pairs in the query string
name_array	The names from the query string (indexed from 1) in the order submitted.
value_array	The values from the query string (indexed from 1) in the order submitted.
reserved	Not used. It is reserved for future use.
array_type (required)	Any PL/SQL index-by table of varchar2 type (e.g., owa.vc_arr).

Example: If you send the following URL, where the *query_string* has duplicate occurrences of the name "x":

```
http://www.acme.com/pls/myDAD/!scott.my_pkg.my_proc?x=a&y=b&x=c
```

The exclamation mark prefix (!) instructs the PL/SQL Gateway to use flexible parameter passing. It invokes procedure `scott.my_pkg.myproc` and passes it the following arguments:

```
num_entries ==> 3
name_array ==> ('x', 'y', 'x');
values_array ==> ('a', 'b', 'c')
reserved ==> ()
```

1.6.3 Large Parameter Passing

The values passed as scalar arguments and the values passed as elements to the index-by table of varchar2 arguments can be up to 32K in size.

For example, when using flexible parameter passing (described in "Flexible Parameter Passing" on page 1-9), each name or value in the *query_string* portion of the URL gets passed as an element of the `name_array` or `value_array` argument to the procedure being invoked. These names or values can be up to 32KB in size.

1.7 File Upload and Download

The PL/SQL Gateway allows you to:

- Upload and download files as raw byte streams without any character set conversions. The files are uploaded into the document table. A primary key is passed to the PL/SQL upload handler routine so that it can retrieve the appropriate table row.
- Specify one or more tables per application for uploaded files so that files from different applications are not mixed together.
- Provide access to files in these tables via a URL format that doesn't use query strings, for example:

  ```
  http://www.acme.com:9000/mysite/pls/docs/cs250/lecture1.htm
  ```

  This is required to support uploading a set of files that have relative URL references to each other.
- Upload multiple files per form submission.
- Upload files into LONG RAW and BLOB types of columns in the document table.

1.7.1 Document Table Definition

You can specify the document storage table on a per DAD basis. The document storage table must have the following definition:

```
CREATE TABLE [table_name] (
    NAME           VARCHAR2(256) UNIQUE NOT NULL,
    MIME_TYPE      VARCHAR2(128),
    DOC_SIZE       NUMBER,
    DAD_CHARSET    VARCHAR2(128),
    LAST_UPDATED   DATE,
    CONTENT_TYPE   VARCHAR2(128),
    [content_column_name] [content_column_type]
```

```
    [ , [content_column_name] [content_column_type]]
);
```

Users can choose the `table_name`. The `content_column_type` type must be either LONG RAW or BLOB.

The `content_column_name` depends on the corresponding `content_column_type`:

- If the content_column_type is LONG RAW, the `content_column_name` must be CONTENT.
- If the content_column_type is BLOB, the `content_column_name` must be BLOB_CONTENT.

An example of legal document table definition is:

```
NAME              VARCHAR(128)   UNIQUE NOT NULL,
MIME_TYPE         VARCHAR(128),
DOC_SIZE          NUMBER,
DAD_CHARSET       VARCHAR(128),
LAST_UPDATED      DATE,
CONTENT_TYPE      VARCHAR(128),
CONTENT           LONG RAW,
BLOB_CONTENT      BLOB ;
```

1.7.1.1 Semantics of the CONTENT column

The contents of the table are stored in a content column. There can be more than one content column in a document table. However, for each row in the document table, only one of the content columns is used. The other content columns are set to NULL.

1.7.1.2 Semantics of the CONTENT_TYPE column

The `content_type` column tracks in which content column the document is stored. When a document is uploaded, the PL/SQL Gateway sets the value of this column to the type name.

For example, if a document was uploaded into the BLOB_CONTENT column, then the `CONTENT_TYPE` column for the document is set to the string 'BLOB'.

1.7.1.3 Semantics of the LAST_UPDATED column

The LAST_UPDATED column reflects a document's creation or last modified time. When a document is uploaded, the PL/SQL Gateway sets the LAST_UPDATED column for the document to the database server time.

If an application then modifies the contents or attributes of the document, it must also update the LAST_UPDATED time.

The PL/SQL Gateway uses the LAST_UPDATED column to check and indicate to the HTTP client (browser) if the browser can use a previously cached version of the document. This reduces network traffic and improves server performance.

1.7.1.4 Semantics of the DAD_CHARSET column

The DAD_CHARSET column keeps track of the character set setting at the time of the file upload. This column is reserved for future use.

1.7.2 Old Style Document Table Definition

For backward capability with the document model used by older releases of WebDB 2.x, the PL/SQL Gateway also supports the following old definition of the document storage table where the CONTENT_TYPE, DAD_CHARSET and LAST_UPDATED columns are not present.

```
/* older style document table definition (DEPRECATED) */
CREATE TABLE [table_name]
(
    NAME         VARCHAR2(128),
    MIME_TYPE    VARCHAR2(128),
    DOC_SIZE     NUMBER,
    CONTENT      LONG RAW
);
```

1.7.3 Relevant Parameters

For each DAD, the following configuration parameters are relevant for file upload or download.

`document_table (document_table_name)`

The `document_table` parameter specifies the table to be used for storing documents when file uploads are performed via this DAD.

Syntax:

```
document_table = [document_table_name]
```

Examples:

```
document_table = my_documents
```

or,

```
document_table = scott.my_document_table
```

1.7.4 document_path (Document Access Path)

The `document_path` parameter specifies the path element to access a document. The `document_path` parameter follows the DAD name in the URL. For example, if the document access path is `docs`, then the URL would look similar to:

```
http://neon/pls/myDAD/docs/myfile.htm
```

The `myDAD` is the DAD name and `myfile.htm` is the file name.

Syntax:

```
document_path = [document_access_path_name]
```

1.7.4.1 document_proc (Document Access Procedure):

The `document_pro` procedure is an application-specified procedure. It has no parameters and processes a URL request with the document access path. The document access procedure calls `wpg_docload.download_file(filename)` to download of a file. It knows the filename based on the URL specification. For example, this can be used by an application to implement file-level access controls and versioning. An example of such an application is shown in "File Download" on page 1-19.

Syntax:

```
document_proc = [document_access_procedure_name]
```

Examples:

```
document_proc = my_access_procedure
```

or,

```
document_proc = scott.my_pkg.my_access_procedure
```

1.7.4.2 upload_as_long_raw

The DAD parameter, `upload_as_long_raw`, configures file uploads based on their file extensions. The value of an `upload_as_long_raw` DAD parameter is a comma separated (,) list of file extensions. Files with these extensions are uploaded by the PL/SQL Gateway into the content column of `long_raw` type in the document table. Files with other extensions are uploaded into the BLOB content column.

The file extensions can be text literals (jpeg, gif, etc.). In addition, an asterisk (*) can be used as a special file extension and matches any file whose extension has not been listed in an `upload_as_long_raw` setting.

Syntax:

```
upload_as_long_raw = [file_extension][,[file_extension]]*
```

`[file_extension]` is an extension for a file (with or without the '.' character, e.g., 'txt' or '.txt') or the wildcard character *.

Examples:

```
upload_as_long_raw = html, txt
upload_as_long_raw = *
```

1.7.5 File Upload

To send files from a client machine to a database, create an HTML page that contains:

- A FORM tag whose *enctype* attribute is set to `multipart/form-data` and whose *action* attribute is associated with a PL/SQL Gateway procedure call, referred to as the "action procedure."
- An INPUT element whose type and name attributes are set to file. The `INPUT type="file"` element enables a user to browse and select files from the file system.

When a user clicks Submit, the following events occur:

1. The browser uploads the file specified by the user as well as other form data to the server.
2. The PL/SQL Gateway stores the file contents in the database in the document storage table. The table name is derived from the document_table DAD setting.
3. The action procedure specified in the *action* attribute of the FORM is run similar to invoking a PL/SQL Gateway procedure without file upload.

The following example shows an HTML form that lets a user select a file from the file system to upload. The form contains other fields that allow the user to provide information about the file.

```
<html>
<head>
<title>test upload</title>
</head>
<body>
 <FORM enctype="multipart/form-data"
action="pls/myDAD/write_info"
method="POST">
<p>Author's Name:<INPUT type="text" name="who">
<p>Description:<INPUT type="text" name="description"><br>
<p>File to upload:<INPUT type="file" name="file"><br>
<p><INPUT type="submit">
</FORM>
</body>
</html>
```

When a user clicks Submit on the form, the browser uploads the file listed in the `INPUT type="file"` element.

The `write_info` procedure then runs. The procedure writes information from the form fields to a table in the database and returns a page to the user. The action procedure does not have to return anything to the user, but it is a good idea to let the user know whether the Submit succeeded or failed.

A sample `write_info` procedure:

```
procedure write_info (
who         in varchar2,
description in varchar2,
file        in varchar2) as
begin
insert into myTable values (who, description, file);
htp.htmlopen;
htp.headopen;
htp.title('File Uploaded');
htp.headclose;
htp.bodyopen;
htp.header(1, 'Upload Status');
htp.print('Uploaded ' || file || ' successfully');
htp.bodyclose;
htp.htmlclose;
end;
```

The filename obtained from the browser is prefixed with a generated directory name to reduce the possibility of name conflicts. The "action procedure" specified in the form renames this name. So, for example, when `/private/minutes.txt` is uploaded, the name stored in the table by the gateway is `F9080/private/minutes.txt`. The application can rename this in the called stored procedure. For example, the application can rename it to `scott/minutes.txt`.

1.7.5.1 Document Parts Upload

When you upload HTML files, they are parsed by the PL/SQL Gateway for other parts (e.g. GIF/JPEG files). These parts details are maintained in a separate table called the documentparts table. The name of this table is based on your document table name. It must have the following signature:

```
create table %YOUR_DOCTABLE_NAME%part
(
 DOCUMENT VARCHAR2(256),
 PART      VARCHAR2(256),
 UPLOADED  CHAR(1),
 [add any more columns you want to add here]
 constraint %YOUR_DOCTABLE_NAME%part_pk primary key( document, part )
)
pctfree 0
```

Where %YOUR_DOCTABLE_NAME% is your document table name (see section Document Table Definition on page 1-12).

Appropriate privileges also must be granted to access this table:

```
grant select, insert, update on %YOUR_DOCTABLE_NAME%part to public
```

When the original HTML document is uploaded, examine the document parts table for any 'parts' that have not been uploaded. Use this information to notify the user. If the 'parts' are not uploaded in the document table, when a user downloads the HTML file, it does not display correctly (e.g. broken images).

1.7.6 Specifying Attributes (Mime Types) of Uploaded Files

In addition to renaming the uploaded file, the stored procedure can alter other file attributes. For example, the form in the example from "File Upload" on page 1-16 could display a field for allowing the user to input the uploaded document's Multipurpose Internet Mail Extension (MIME) type.

The MIME type can be received as a parameter in `write_info`. The document table would then store the mime type for the document instead of the default mime type that is parsed from the multipart form by the PL/SQL Gateway when uploading the file.

1.7.7 Uploading Multiple Files

To send multiple files in a single submit, the upload form must include multiple <INPUT type="file" name="file"> elements. If more than one file INPUT element defines `name` to be of the same name, then the action procedure must declare that parameter name to be of type owa.vc_arr. The names defined in the file INPUT elements could also be unique, in which case, the action procedure must declare each of them to be of varchar2. For example, if a form contained the following elements:

```
<INPUT type="file" name="textfiles">
<INPUT type="file" name="textfiles">
<INPUT type="file" name="binaryfile">
```

As a result, the action procedure must contain the following parameters:

```
procedure handle_text_and_binary_files(textfiles IN owa.vc_arr,
binaryfile IN varchar2).
```

1.7.8 File Download

After you have sent files to the database, you can download them, delete them from the database, and read and write their attributes.

To download a file, create a stored procedure without parameters that calls `wpg_docload.download_file` (file_name) to initiate the download.

The HTML page presented to the user simply has a link to a URL which includes the Document Access Path and specifies the file to be downloaded.

For example, if the DAD specifies that the Document Access Path is docs and the Document Access Procedure is webview.process_download, then the webview.process_download procedure is called when the user clicks on the URL:

```
http://www.acme:9000/pls/webview/docs/myfile.htm
```

An example implementation of process_download is:

```
procedure process_download is
v_filename varchar2(255);
begin
```

```
  -- getfilepath() uses the SCRIPT_NAME and PATH_INFO cgi
  -- environment variables to construct the full pathname of
  -- the file URL, and then returns the part of the pathname
  -- following '/docs/'
  v_filename := getfilepath;
  select name into v_filename from plsql_gateway_doc
  where UPPER(name) = UPPER(v_filename);
  -- now we call docload.download_file to initiate
  -- the download.
  wpg_docload.download_file(v_filename);
exception
  when others then
v_filename := null;
end process_download;
```

Any time you call `wpg_docload.download_file(filename)` from a procedure running in the Gateway, a download of the file *`filename`* is initiated. However, when a file download is initiated, no other HTML (produced via HTP interfaces) generated by the procedure, is passed back to the browser.

The PL/SQL Gateway looks for the filename in the document table. There must be a unique row in the document table whose NAME column matches the filename. The PL/SQL Gateway generates HTTP response headers based on the information in the MIME_TYPE column of the document table. The `content_type` column's value determines which content columns the document's content comes from. The contents of the document are sent as the body of the HTTP response.

1.7.9 Direct BLOB Download

You can also download contents that are stored as a Binary Large Object (BLOB) data type.

1. Create a stored procedure that calls wpg_docload.download_file(blob) where blob is of data type BLOB. Since the PL/SQL Gateway has no information about the contents in the BLOB, you must supply them.
2. Setup the Content-Type and other headers.

   Example: The following procedure uses the name from the argument to select a BLOB from a table and initiates the Direct BLOB download:

   ```
   procedure download_blob(varchar2 name) is
   myblob blob;
   begin
   ```

a. Select the blob out of mytable using the name argument

```
select blob_data into myblob from mytable where blob_name = name;
```

b. Setup headers which describes the content

```
owa_util.mime_header('text/html', FALSE);
htp.p('Content-Length: ' || dbms_lob.get_length(myblob));
owa_util.http_header_close;
```

c. Initiate Direct BLOB download

```
wpg_docload.download_file(myblob);
end;
```

The structure of the mytable table:

```
create table mytable
(
blob_name varchar2(128),
blob_data blob
);
```

3. The HTML page presented to the user has a link to a URL that calls this stored procedure with the correct argument(s).
4. When a Direct BLOB download is initiated, no other HTML (produced via the HTP interface) generated by the procedure is passed back to the browser.

1.8 Path Aliasing (Direct Access URLs)

Path Aliasing enables applications using the PL/SQL Gateway to provide direct reference to its objects using simple URLs. The PL/SQL Gateway allows you to directly access documents within an application using the document access path and a document access procedure. For example, the `docs` keyword in the URL below tells the PL/SQL Gateway that this request is for document access.

`http://<HostName>[:Port]/<DADName>/docs/<FolderName/Document>`

The above assumes that the Document Access Path is `docs`.

Path Aliasing provides the equivalent function by allowing means of direct access to application objects other than documents. Two fields in Database Access Descriptor's configuration information support path aliasing:

- Path Alias
- Path Alias Procedure

If the PL/SQL Gateway encounters in an incoming URL the keyword entered in the **Path Alias** field, it invokes the procedure entered in the **Path Alias Procedure** field.

For example, if the incoming URL is

```
http://www.acme.com:9000/portal_DAD/URL/path_alias_URL
```

and the Path Alias is URL, the PL/SQL Gateway invokes the **Path Alias Procedure,** passing everything after the keyword URL to the invoked procedure.

Applications that use path aliasing must implement the **Path Alias Procedure**. The procedure receives the rest of the URL (path_alias_URL) after the keyword, URL, as a single parameter, and is therefore responsible and also fully capable of dereferencing the object from the URL.

Although there is no restriction on the name and location for this procedure, it can accept only a single parameter, p_path, with the datatype varchar2.

1.9 Common Gateway Interface (CGI) Environment Variables

The OWA_UTIL package provides an API to get the values of CGI environment variables, which serve to provide context to the procedure being executed via the PL/SQL Gateway. Although the PL/SQL Gateway is not operated through CGI, the PL/SQL application invoked from the PL/SQL Gateway can access these CGI environment variables. The following is a list of the available CGI Environment Variables:

Table 1–5 CGI Environment Variables

CGI Environment Variables	
AUTHORIZATION	DAD_NAME
DOC_ACCESS_PATH	DOCUMENT_TABLE (refer to "document_table (document_table_name)"
HTTP_ACCEPT	HTTP_ACCEPT_ENCODING
HTTP_ACCEPT_CHARSET	HTTP_ACCEPT_LANGUAGE
HTTP_COOKIE	HTTP_HOST
HTTP_PRAGMA	HTTP_REFERER
HTTP_USER_AGENT	PATH_ALIAS
PATH_INFO	REMOTE_ADDR
REMOTE_HOST	REMOTE_USER (refer to "REMOTE_USER CGI Environment Variable" on page 2-4)

Table 1–5 CGI Environment Variables

CGI Environment Variables	
REQUEST_CHARSET (refer to "REQUEST_ CHARSET CGI environment variable" on page 1-24)	REQUEST_IANA_CHARSET
REQUEST_METHOD	REQUEST_PROTOCOL
SCRIPT_NAME	SCRIPT_PREFIX
SERVER_NAME	SERVER_PORT
SERVER_PROTOCOL	

A PL/SQL application can get the value of a CGI environment variable using the owa_util.get_cgi_env interface.

Syntax:

```
owa_util.get_cgi_env(param_name in varchar2) return varchar2;
```

param_name is the name of the CGI environment variable. param_name is case-insensitive.

1.9.1 Adding and Overiding CGI Environment Variables

The `cgi_env_list` DAD parameter is a comma separated list of name and value pairs which can override any environment variables or add new ones. If the name is one of the original environment variables (as listed in "Common Gateway Interface (CGI) Environment Variables" on page 1-22), that environment variable is overridden with the given value. If the name is not in the original list, a new environment variable is added into the list with that same name and value given in the parameter.

If no value is specified for the parameter, then the value is obtained from the Oracle HTTP Server. With Apache, you can pass the DOCUMENT_ROOT CGI Environment variable by specifying:

```
cgi_env_list=DOCUMENT_ROOT
```

Access `cgi_env_list` through the PL/SQL Gateway configuration file (`wdbsvr.app`). This configuration file describes settings for the PL/SQL Gateway module. The location of your Oracle9*i* Application Server installation is *<ORACLE_HOME>*. For UNIX, the configuration file is located at:

<*ORACLE_HOME*>/Apache/modplsql/cfg/wdbsvr.app

For NT, it is located at:

<*ORACLE_HOME*>\Apache\modplsql\cfg\wdbsvr.app

Example 1:

```
cgi_env_list=SERVER_NAME=myhost.mycompany.com, REMOTE_
USER=testuser
```

This example overrides the SERVER_NAME and the REMOTE_USER CGI environment variables with the given values since they are part of the original list.

Example 2:

```
cgi_env_list=MYENV_VAR=testing, SERVER_NAME=,REMOTE_USER=user2
```

This example overrides the SERVER_NAME and the REMOTE_USER variables. The SERVER_NAME variable is deleted since there is no value given to it. A new environment variable called MYENV_VAR is added since it is not part of the original list. It is assigned the value of "testing".

1.9.2 NLS_LANG

For PL/SQL Gateway mod_plsql, the National Language Support variable (NLS_LANG) can be set either as an environment variable or at the DAD level. Refer to "DAD Settings Accessible through the Configuration File" on page 4-8 for more information. The following restrictions apply:

- The NLS_LANG parameter of the database must match that of the Oracle HTTP Server *powered by Apache*, or
- The NLS_LANG parameter of the database and Oracle HTTP Server *powered by Apache*, must be of fixed character width and both must be the same size.

1.9.2.1 REQUEST_CHARSET CGI environment variable

Every request to the PL/SQL Gateway is associated with a DAD. The CGI environment variable REQUEST_CHARSET is set as follows:

- The REQUEST_CHARSET is set to the default character set in use.
  - For the embedded Gateway, this is the database's default character set.
  - For the Gateway deployed in the middle-tier (as part of the Oracle HTTP Server *powered by Apache*), this is the character set information derived from the NLS_LANG environment variable.However, if the DAD level NLS_

LANG parameter is set, that is used to derive the character set information instead.

The PL/SQL application can access this information via a function call of the form:

```
owa_util.get_cgi_env('REQUEST_CHARSET');
```

1.9.2.2 REQUEST_IANA_CHARSET CGI environment variable

This is the IANA (Internet Assigned Number Authority) equivalent of the REQUEST_CHARSET CGI environment variable. IANA is an authority that globally coordinates the standards for charsets used on the Internet.

2

Securing Applications through the PL/SQL Gateway

There are essentially two types of PL/SQL Gateway security. One is authentication, which establishes who the user(s) is. The second is protection, which determines the privileges of the user(s).

2.1 Authenticating Users

The PL/SQL Gateway provides different levels of authentication in addition to those provided by the Web Server. The Web server protects documents, virtual paths and so forth, while the PL/SQL Gateway protects users logging into the database or running a PL/SQL Web application.

You can enable different authentication modes using the **Authentication Mode** parameter on the Gateway Configuration pages.

- **Basic** - authentication is performed using Basic HTTP Authentication. Most applications use Basic Authentication.
- **Global Owa** - authentication is performed in the schema containing the PL/SQL Web Toolkit packages.
- **Custom Owa** - authentication is performed using packages and procedures in the user's schema, or if not found, in the schema containing the PL/SQL Web Toolkit packages.
- **PerPackage** - authentication is performed using packages and procedures in the user's schema
- **Single Sign-On** - authentication is performed using the Oracle Single Sign-On feature of the Login Server. Use this mode only if your application works with the Login Server (Oracle Portal only).

2.1.1 Basic (Database Controlled Authentication)

The PL/SQL Gateway supports authentication at the database level. It uses HTTP Basic Authentication but authenticates credentials by using them to attempt to log on to the database. Authentication is verified against a user database account, using user names and passwords that are either:

- stored in the DAD. The end user is not required to log in. This method is useful for Web pages that provide public information.
- provided by the users via the browser's Basic HTTP Authentication dialog box. The end user must provide a username and password in the dialog box.

Oracle Application Server (OAS) Basic Authentication Mode

OAS has a different mechanism for the Basic Authentication Mode. The username and password must be stored in the DAD. OAS uses HTTP Basic Authentication where the credentials are stored in a password file on the file system. Authentication is verified against the users listed in that file.

The PL/SQL Gateway supports OAS Basic Authentication. The Oracle HTTP Server that comes with Oracle9*i* Application Server can authenticate users' credentials against a password file on the file system. This functionality is provided by a module called `mod_auth`.

2.1.1.1 Deauthentication

The PL/SQL Gateway allows users to log off (clear HTTP authentication information) programatically through a PL/SQL procedure without having to exit all browser instances. This feature is supported on Netscape 3.0 or higher and Internet Explorer. On other browsers, the user may have to exit the browser to deauthenticate.

Another method of deauthentication is to add `/logmeoff` after the DAD in the URL, for example:

```
http://myhost:2000/pls/myDAD/logmeoff
```

2.1.2 Global OWA, Custom OWA, and Per Package (Custom Authentication)

Custom authentication enables applications to authenticate users within the application itself, not at the database level. Authorization is performed by invoking a user-written authorization function.

Implementing the authorize function

Custom authentication uses a static username/password that is stored in the DAD. It cannot be combined with dynamic username/password authentication.

The syntax of the authorize function is:

```
function authorize return boolean;
```

To enable custom authentication:

1. Set the level of authentication on the DAD Configuration page.
2. Implement the authorize function.

The PL/SQL Gateway uses the username/password provided in the DAD to log into the database. Once the login is complete, authentication control is passed to the application. Application-level PL/SQL hooks (callback functions) are then called. The implementations for these callback functions are left to the application developers. The return value of the callback function determines if the authentication succeeded or failed: if the function returns TRUE, authentication succeeded. If it returns FALSE, authentication failed and code in the application is not executed.

You can place the authentication function in different locations, depending on when it is to be invoked:

- To invoke the same authentication function for all users and procedures, choose **Global OWA** in the **Authentication Mode** list on the DAD Configuration Page. Then, implement the `owa_custom.authorize` function in the schema that contains the PL/SQL Web Toolkit, which is SYS.
- To invoke a different authentication function for each user and for all procedures, choose **Custom OWA** in the **Authentication Mode** list on the DAD Configuration Page. Then implement the `owa_custom.authorize` function in each user's schema. For users who do not have that function, the `owa_custom.authorize` function in the PL/SQL Web Toolkit package schema is invoked instead.
- To invoke the authentication function for all users but only for procedures in a specific package or for anonymous procedures, choose **Per Package** in the **Authentication Mode** list on the DAD Configuration Page. Then, implement the authorize function in that package in each user's schema. If the procedure is

not in a package, then the anonymous authorize function is called instead. The following table summarizes the parameter values:

Table 2–1 Custom Authentication Modes and Callback Functions

Mode	Access control scope	Callback function
Global OWA	All packages	owa_custom.authorize in the OWA package schema
Custom OWA	All package	owa_custom.authorize in the user's schema, or, if not found, in the OWA package schema
Per Package	Specified package	packageName.authorize in the user's schema, or anonymous.authorize is called.

2.1.3 REMOTE_USER CGI Environment Variable

The REMOTE_USER CGI environment variable has different username values depending on the authentication mode. The following list explains how this variable is derived:

Basic: If the username and password is stored in the DAD, then it is same username. If the username and password is empty in the DAD, then it is the username which the user entered in the HTTP authentication dialog box.

- **OAS's Basic:** The username that the user entered in the HTTP Authentication dialog box.

Global OWA, Custom OWA, and PerPackage: The username that the user entered in the HTTP authentication dialog box.

Single Sign-On: The username of the lightweight user that was authenticated by the login server.

2.2 Protecting the PL/SQL Procedures Granted to PUBLIC

With the different levels of authentication, you must protect the execution of the PL/SQL procedures granted to PUBLIC in the database. These procedures (in the dbms_% packages, utl_% packages, and all packages under the SYS schema) pose a security hole when they are executed through the Web browser. Such packages are intended only for the PL/SQL application developer. Some procedures in the dbms_% packages allow access to sensitive information.

The PL/SQL Gateway provides a DAD parameter to protect the execution of these PL/SQL packages and other application specific packages.

The DAD parameter is called `exclusion_list`. This is a comma separated list of procedure/package/schema names which are forbidden to be directly executed from a browser. Each string in the list is case-insensitive and can accept wildcards such as *, ?, and [a-z]. If this parameter is not specified, the default list is `sys.*, dbms_*, utl_*, owa_util.showsource`, and `owa_util.cellsprint`. Setting this parameter to #NONE# allows all procedures to be accessed. If the user changes the value of this parameter, then there are no system defaults. Therefore, to add more procedures, you have to add the original list as well.

Access `exclusion_list` through the PL/SQL Gateway configuration file (`wdbsvr.app`). The location of your Oracle9*i* Application Server installation is <*ORACLE_HOME*> .For UNIX, the configuration file is located at:

<*ORACLE_HOME*>/Apache/modplsql/cfg/wdbsvr.app

For NT, it is located at:

<*ORACLE_HOME*>\Apache\modplsql\cfg\wdbsvr.app

Example 1:

```
exclusion_list=testschema.testpkg.*,sys.*,dbms_*,pkg?.*
```

This example protects all procedures in testschema.testpkg package, all procedures in the sys schema, all procedures in packages that match dbms_, and all procedures in packages that start with pkg and one character that can be a wildcard.

Example 2:

```
exclusion_list=#NONE#
```

This example disables all protection. This is recommended for debugging purposes only. This is not recommended for production environments.

2.3 Protecting the Administration pages

DBAs responsible for granting privileges to the DAD Administration pages need to protect these pages from public access. You can protect the admin pages through

Basic Authentication or through "Protecting the Admin pages through the Oracle Portal (Single Sign-On)" on page 2-8 if you are using Oracle Portal.

Note: Verify that the DAD parameter error_style is not set to "GatewayDebug" in a production environment. This setting produces information as specified by the "Gateway" setting and server configuration related information. This mode is only meant for debugging purposes. Displaying server internal variables could produce a security risk.

2.3.1 Protecting the Admin pages through Basic Authentication

Using Basic Authentication, the PL/SQL Gateway uses the DAD entered in the URL to connect to a database and authenticate the user's credentials. The user enters the username and password information into the Basic HTTP Authentication dialog box from the browser to gain access to the Admin pages.

To enable this type of protection, edit the configuration file (wdbsvr.app) as follows:

1. Open the file named `wdbsvr.app`. The location of your Oracle9*i* Application Server installation is *<ORACLE_HOME>*. For UNIX, the configuration file is located at:

   <ORACLE_HOME>`/Apache/modplsql/cfg/wdbsvr.app`

   For NT, it is located at:

   <ORACLE_HOME>`\Apache\modplsql\cfg\wdbsvr.app`

2. In the WVGATEWAY section, located at the top of the file, locate the administrators parameter.

3. Enter valid usernames for the people who need to access the Admin pages. This is a comma separated list.

   Example 1:

   ```
   administrators=adminuser1,adminuser2
   ```

   Only adminuser1 and adminuser2 can access the Admin Pages.

   Example 2:

   ```
   administrators=all
   ```

Everyone can access the Admin Pages. This is only recommended for development environments.

4. Verify the admindad parameter is set to blank as shown below:

```
admindad=
```

5. Create a new DAD which admin page authentication will use to connect to a database. Leave the username and password parameters blank. Set the enablesso parameter to No. Set the other parameters to blank or comment them out.

Example:

```
[DAD_MyDADName]
connect_string   =  my_connect_string
password   =
username   =
default_page   =
document_table   =
document_path   =
document_proc   =
upload_as_long_raw   =
upload_as_blob   =
name_prefix   =
always_describe   =
after_proc   =
before_proc   =
reuse   =  yes
pathalias   =
pathaliasproc   =
enablesso   =  No
;sncookiename   =
;stateful   =
;custom_auth   =
;response_array_size   =
;
```

Now, only usernames that appear in the administrators parameter list have permission to access Admin pages.

6. Type the following URL:

```
http://hostname:port/pls/MyDADName/admin_/gateway.htm
```

You are then prompted by a Basic HTTP Authentication dialog box to enter your username and password.

- If the credentials are verified against the MyDADName DAD and the username is part of the administrators list, you can access the Admin Pages.
- If the credentials are not verified against the MyDADName DAD or the username is not in the administrators list, you cannot access the Admin pages. An error message is displayed.

2.3.2 Protecting the Admin pages through the Oracle Portal (Single Sign-On)

DBAs responsible for granting privileges to the DAD Administration pages need to protect these pages from public access. Only the DBA or users with DBA-level privilege in Oracle Portal can access the DAD pages if the configuration file is edited as follows:

1. Open the PL/SQL Gateway configuration file named `wdbsvr.app`. The location of your Oracle9*i* Application Server installation is *<ORACLE_HOME>.* For UNIX, the configuration file is located at:

   <ORACLE_HOME>`/Apache/modplsql/cfg/wdbsvr.app`

   For NT, it is located at:

   <ORACLE_HOME> \ Apache \ modplsql \ cfg \ wdbsvr.app

2. In the `[WVGATEWAY]` section, typically located at the top of the file, locate the `admindad` parameter.

3. Enter a valid DAD name for a schema that has Single Sign-On enabled (`enablesso=yes`). Usually this name is set to the name of the DAD in which your Oracle Portal objects are installed. By default, the name is `portal30`.

   In the `wdbsvr.app` file, the Oracle Portal 3.0 gateway security parameters are displayed similar to the following:

   ```
   administrators = all
   adminPath = /admin_/
   admindad = portal30
   ```

   The name of the schema containing your Oracle Portal installation is `portal30`.

4. Replace `portal30` with the name of your Oracle Portal DAD.

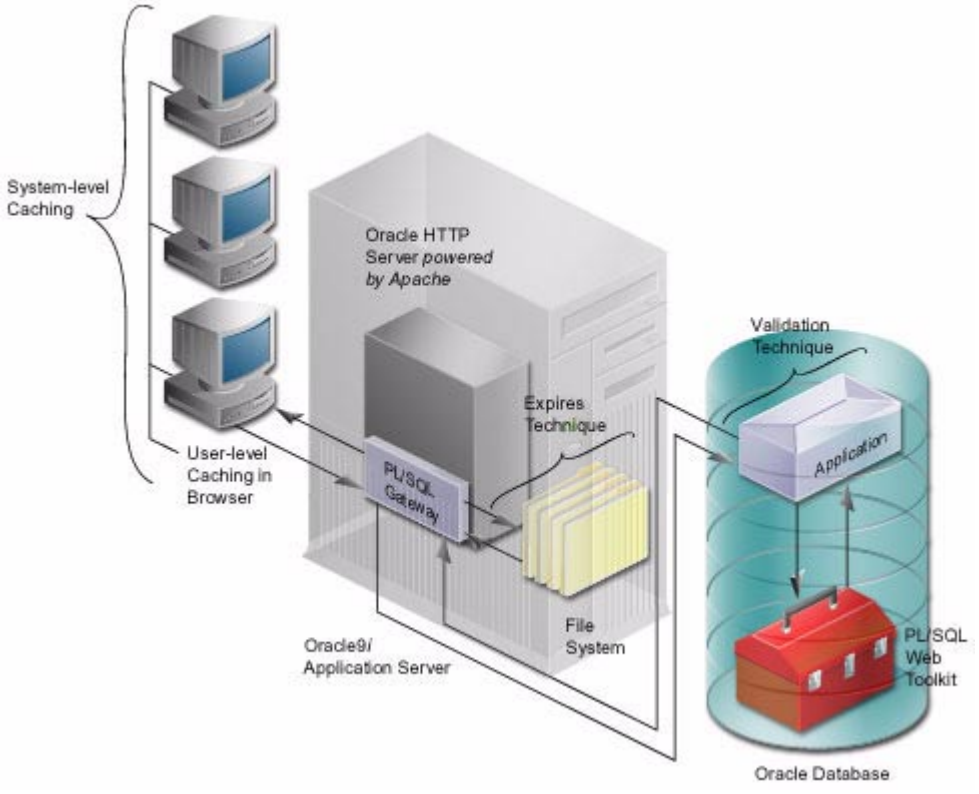
   When a user tries to access the DAD administration pages, authentication is performed through the Single Sign-On.

   - If the user is authorized to access these pages, the main DAD configuration page is displayed.
   - If the user is not authorized, the Single Sign-On page prompts for a user name and password. If this information does not authorize the user to access these pages, an error message is displayed.

3

Caching

Caching can improve performance of your PL/SQL Web applications. You can cache Web content generated by PL/SQL procedures in the middle-tier and decrease the database workload.



3.1 The PL/SQL Gateway Cache

Refer to the Caching Overview diagram above during the discussion of caching. It illustrates the different techniques used in caching. For example:

- Validation Technique - An application asks the server if the page has been modified since it was last presented.
- Expires Technique - Based upon a specific time period the PL/SQL application determines if the page will be cached or should be generated again.
- System or User-level Caching - Valid whether you are using the Validation Technique or the Expires Technique. The level of caching is determined by whether a page is cached for a particular user or for every user in the system.

These techniques and levels are implemented using owa_cache packages located inside the PL/SQL Web Toolkit, represented in the diagram by the toolkit in the database.

3.2 Validation Technique

In general, the validation technique basically asks the server if the page has been modified since it was last presented. If it has not been modified, the cached page will be presented to the user. If the page has been modified, a new copy will be retrieved, presented to the user and then cached.

There are two methods which use the Validation Technique, Last-Modified method and the Entity Tag method. The next two sections show how these techniques are used in the HTTP protocol. Although the PL/SQL Gateway does not use the HTTP protocol, many of the same are principles are used.

3.2.1 Last-Modified

Using the HTTP protocol, when a Web page is generated, it contains a **Last-Modified** Response Header. This header indicates the date (relative to the server) of the content that was requested. Browsers save this date information along with the content. When subsequent requests are made for the URL of the Web page, the browser:

1. Determines if it has a cached version.
2. Extracts the date information.
3. Generates the Request Header **If-Modified-Since**.
4. Sends the request the server.

Cache-enabled servers look for the **If-Modified-Since** header and compare it to their content's date. If the two match, an HTTP Response status header such as "HTTP/1.1 304 Not Modified" is generated, and no content is streamed. Upon receipt of this status code, the browser can reuse its cache entry because it has been validated.

If the two don't match, an HTTP Response header such as "HTTP/1.1 200 OK" is generated and the new content is streamed, along with a new **Last-Modified Response h**eader. Upon receipt of this status code, the browser must replace its cache entry with the new content and new date information.

3.2.2 Entity Tag Method

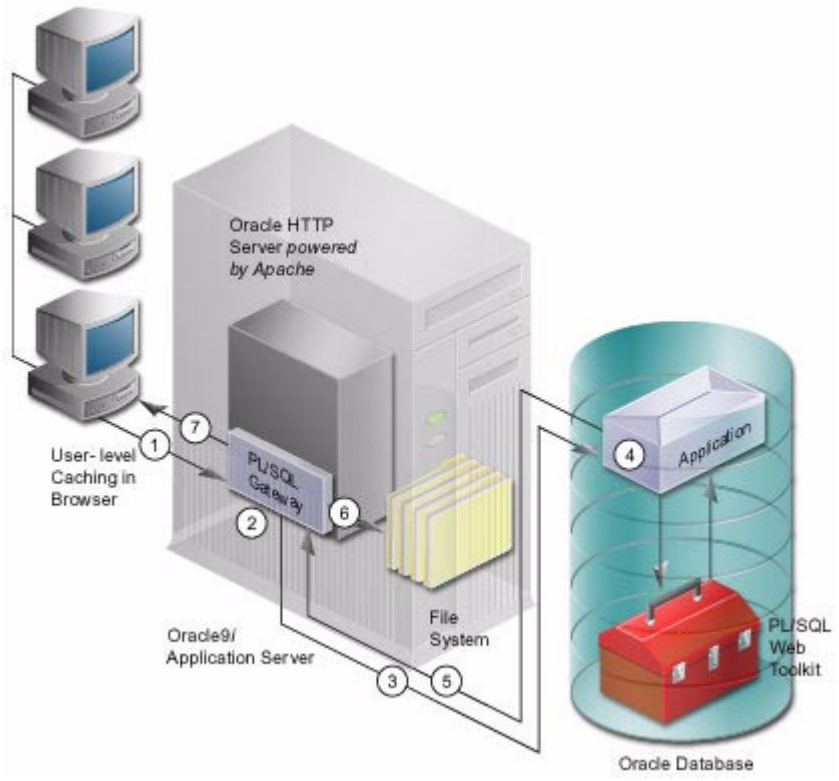
Another validation method provided by the HTTP protocol is the **ETag** (Entity Tag) Response and Request header. The value of this header is a string that is opaque to the browser. Servers generate this string based on their type of application. This is a more generic validation method than the **If-Modified-Since** header, which can only contain a date value.

The **ETag** method works very similar to the Last Modified method. Servers generate the ETag as part of the Response Header. The browser stores this opaque header value along with the content that is steamed back. When the next request for this content arrives, the browser passes the **If-Match** header with the opaque value that it stored to the server. Because the server generated this opaque value, it is able to determine what to send back to the browser. The rest is exactly like the **Last-Modified** validation method as described above.

3.2.3 Using the Validation Technique for PL/SQL Gateway

Using HTTP validation caching as a framework, the following is the Validation Model for the PL/SQL Gateway.

PL/SQL applications that want to control the content being served should use this type of caching. This technique offers some moderate performance gains. One example of this would be an application that serves dynamic content that can change at any given time. In this case, the application needs full control over what is being served. Validation caching always asks the application whether the cached content is stale or not before serving it back to the browser.



1. The Oracle HTTP Server receives a PL/SQL procedure request from a client server. The Oracle HTTP Server routes the request to the PL/SQL Gateway.
2. PL/SQL Gateway prepares the request.
3. PL/SQL Gateway invokes the PL/SQL procedure in the application. The PL/SQL Gateway passes the usual Common Gateway Interface (CGI) environment variables to the application.
4. The PL/SQL procedure generates content to pass back. If the PL/SQL procedure decides that the generated content is cacheable, it calls the owa_cache procedure from the PL/SQL Web Toolkit to set the tag and cache level:

```
owa_cache.set_cache(p_etag, p_level);
```

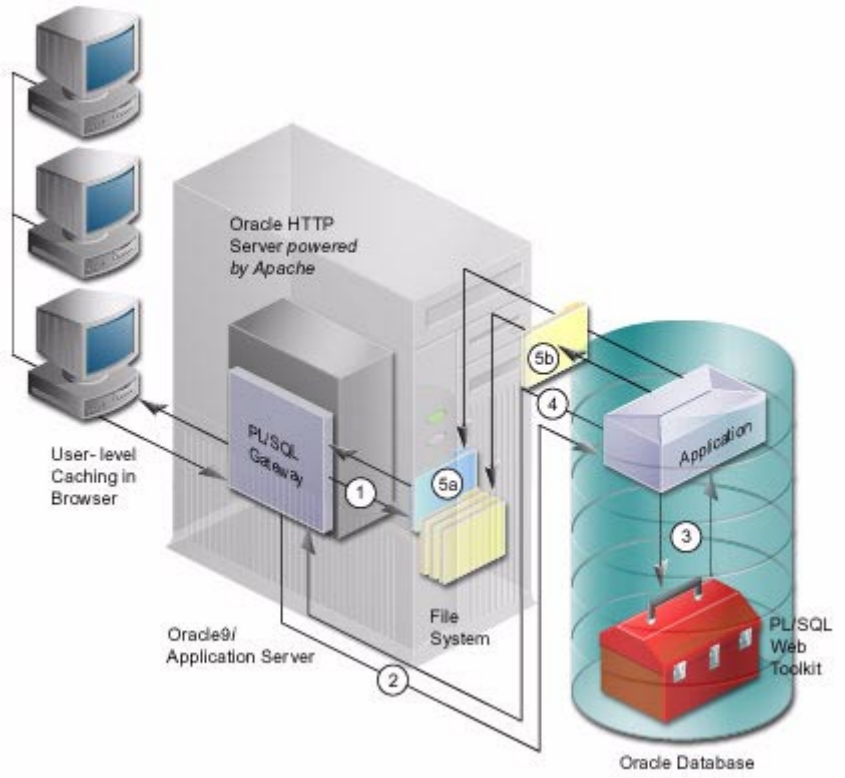
Table 3–1 Validation Model Parameters

Parameter	Description
`set_cache` procedure	Sets up the headers to notify mod_plsql that the content being streamed back can be cached. Then, the mod_plsql caches the content on the local file system along with the tag and caching level information as it is streamed back to the browser.
p_etag	The string that the procedure generates to tag the content.
p_level	The caching level ("SYSTEM" for system level or "USER" for user level).

5. The HTML is returned to the PL/SQL Gateway.
6. The PL/SQL Gateway stores the cacheable content in its file system for the next request.
7. The Oracle HTTP Server sends the response to the client browser.

3.2.4 Second Request using the Validation Technique

Using the same validation model explained above, a second request is made by the client browser for the same PL/SQL procedure.



1. PL/SQL Gateway detects that it has a cached content for the request.
2. The PL/SQL Gateway forwards the same tag and caching level information (from the first request) to the PL/SQL procedure as part of the CGI environment variables.
3. The PL/SQL procedure uses these caching CGI environment variables to check if the content has changed. It does so by calling the following `owa_cache` functions from the PL/SQL Web Toolkit:

   ```
   owa_cache.get_etag;

   owa_cache.get_level;
   ```

   These owa functions get the tag and caching level.

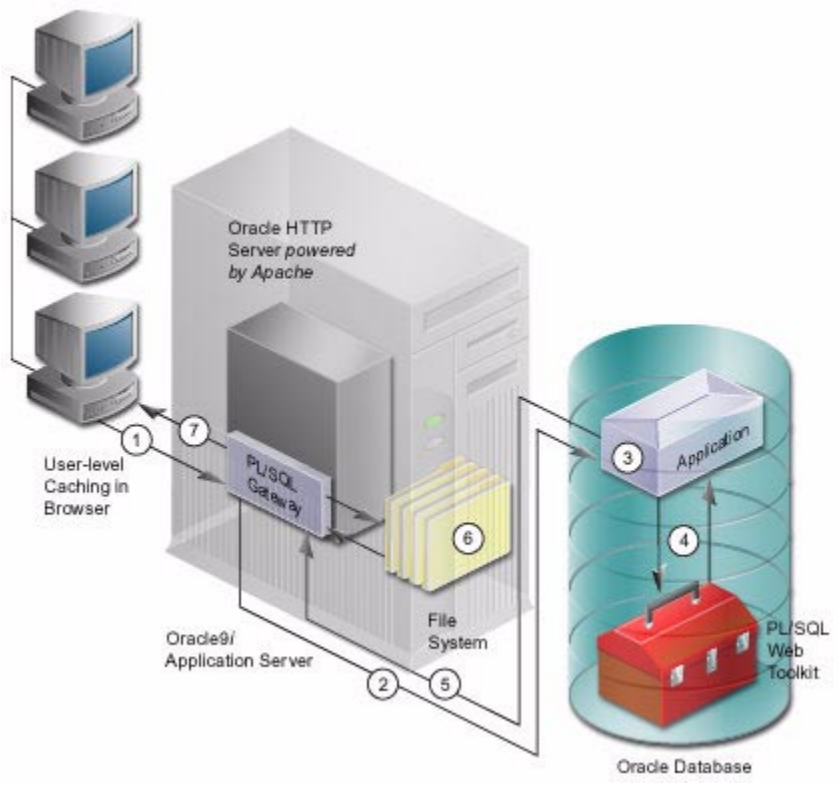
4. The application sends the caching information to the PL/SQL Gateway.
5. Based on that information determines whether the content needs to be regenerated or can be served from the cache.
   a. If the content is still the same, the procedure calls the `owa_cache.set_not_modified` procedure and generates no content. This PL/SQL Gateway to use its cached content. The cached content is directly streamed back to the browser.
   b. If the content has changed, it generates the new content along with a new tag and caching level. PL/SQL Gateway replaces its stale cached copy with a new one and updates the tag and caching level information. The newly generated content is streamed back to the browser.

3.3 Using the Expires Technique

In the validation model, the PL/SQL Gateway always asks the PL/SQL procedure if it can serve the content from the cache. In the expires model, the procedure preestablishes the content validity period. Therefore, the PL/SQL Gateway can serve the content from its cache without asking the procedure. This further improves performance because no interaction with the database is required.

This caching technique offers the best performance. Use if your PL/SQL application is not sensitive to serving stale content. One example of this is an application that generates news daily. The news can be set to be valid for 24 hours. Within the 24 hours, the cached content is served back without contacting the application. This is essentially the same as serving a file. After 24 hours, the PL/SQL Gateway will again fetch new content from the application.

Assume the same scenario described above for the Validation model, except the procedure uses the Expires model for caching.



1. The Oracle HTTP Server receives a PL/SQL Server Page request from a client server. The Oracle HTTP Server routes the request to the PL/SQL Gateway.
2. The request is forwarded by PL/SQL Gateway to the Oracle Database.
3. The PL/SQL Gateway invokes the PL/SQL procedure in the application and passes the usual Common Gateway Interface (CGI) environment variables to the application.
4. The PL/SQL procedure generates content to pass back. If the PL/SQL procedure decides that the generated content is cacheable, it calls the owa_cache procedure from the PL/SQL Web Toolkit to set the validity period and cache level:

```
owa_cache.set_expires(p_expires, p_level);
```

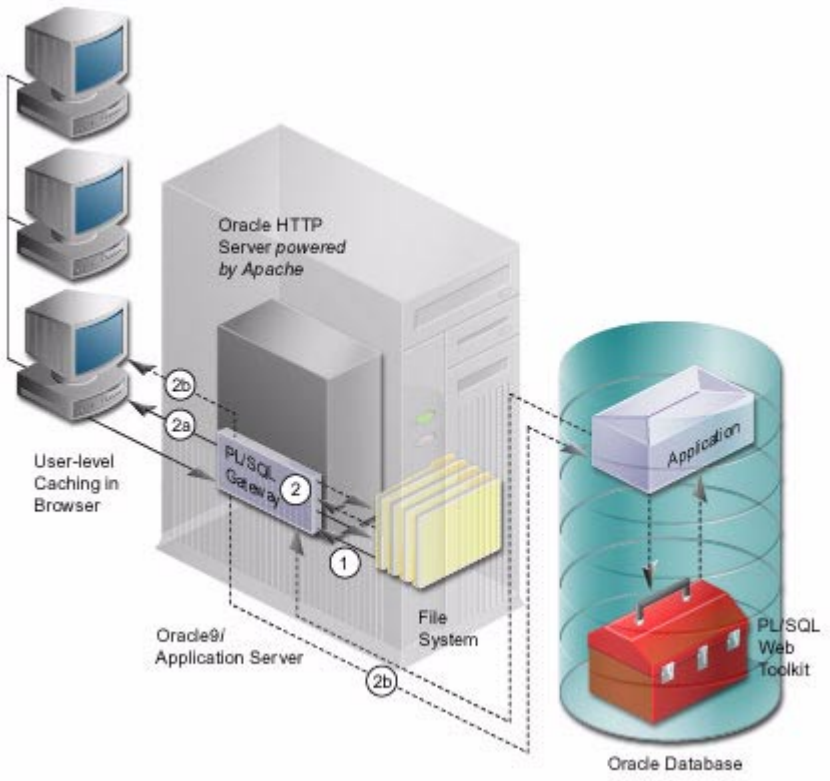
Table 3–2 Expires Model Parameters

Parameter	Description
`set_expires` procedure	Sets up the headers to notify mod_plsql that Expires caching is being used. Mod_plsql then caches the content to the file system along with the validity period and caching level information.
p_expires	Number of minutes that the content is valid.
p_level	Caching level.

5. The HTML is returned to the PL/SQL Gateway.
6. The PL/SQL Gateway stores the cacheable content in its file system for the next request.
7. The Oracle HTTP Server send s the response to the client browser.

3.3.1 Second Request using the Expires Technique

Using the same expires model explained above, a second request is made by the client browser for the same PL/SQL procedure.



1. The PL/SQL Gateway detects that it has a cached copy of the content that is expires-based.
2. The PL/SQL Gateway checks the content's validity by taking the difference between the current time and the time this cache file was created.
   a. If this difference is within the validity period, the cached copy is still fresh and will be used without any database interaction. The cached content is directly streamed back to the browser.
   b. If the difference is not within the validity period, the cached copy is stale. The PL/SQL Gateway invokes the PL/SQL procedure and generates new content. The procedure then decides whether to use expires-based caching again. If so, it also determines the validating period for this new content. The newly generated content is streamed back to the browser.

3.4 System- and User-level Caching

A PL/SQL procedure determines whether generated content is system-level content or user-level. This helps the PL/SQL Gateway cache to store less redundant files if more than one user is looking at the same content. It decides this by:

- For **system-level** content, the procedure passes the string "SYSTEM" as the caching level parameter to the `owa_cache` functions (`set_cache` for validation model or `set_expires` for expires model). This is for every user that shares the cache.
  By using system-level caching, you can save both space in your file system and time for all users in the system. One example of this would be an application that generates content that is intended for everybody using the application. By caching the content with the system-level setting, only one copy of the content is cached in the file system. Furthermore, every user on that system benefits since the content is served directory from the cache.
- For **user-level** content, it passes the string "USER" as the parameter for the caching level. This is for a specific user that is logged in. The stored cache is unique for that user. Only that user can use the cache.The type of user is determined by the authentication mode. Refer to the table below for the different types of users. For more information, refer to "Securing Applications through the PL/SQL Gateway" on page 2-1.

Table 3–3 Type of user determined by Authentication Mode

Authentication Mode	Type of User
Single Sign On (SSO)	Lightweight user
Basic	Database user
Custom	Remote user

For example, if no user customizes a PL/SQL Web application, then the output can be stored in a system-level cache. There will be only one cache copy for every user on the system. User information is not used since the cache can be used by multiple users.

However, if a user customizes the application, a user-level cache is stored for that user only. All other users still use the system level cache. For a user-level cache hit, the user information is a criteria. A user-level cache always overrides a system-level cache.

3.4.1 PL/SQL Web Toolkit functions (owa_cache package)

Your decision whether to use the Validation technique or the Expires technique determines which owa_cache functions to call.

The owa_cache package contains procedures to set and get special caching headers and environment variables. These allow developers to use the PL/SQL Gateway cache more easily. This package should already be installed in your database.

These are the primary functions to call:

Table 3–4 Primary owa_cache functions

owa Functions	Purpose
`owa_cache.set_cache (p_etag IN varchar2, p_level IN varchar2)`	Validation Model - Sets up the headers. ■ p_etag parameter tags the generated content. ■ p_level parameter is the caching level to use.
`owa_cache.set_not_ modified`	Validation Model - Sets up the headers to notify mod_plsql to use the cached content. Only used when a validation -based cache hit occurs.
`owa_cache.get_level`	Validation Model - Gets the caching level, "USER" or "SYSTEM". Returns null if the cache is not hit.
`owa_cache.get_etag`	Validation Model - Gets the tag associated with the cached content. Returns null if the cache is not hit.
`owa_cache.set_expires( p_expires IN number, p_level IN varchar2)`	Expires Model - Sets up the headers. ■ p_expires parameter is the number of minutes the content is valid. ■ p_level parameter is the caching level to use.

4

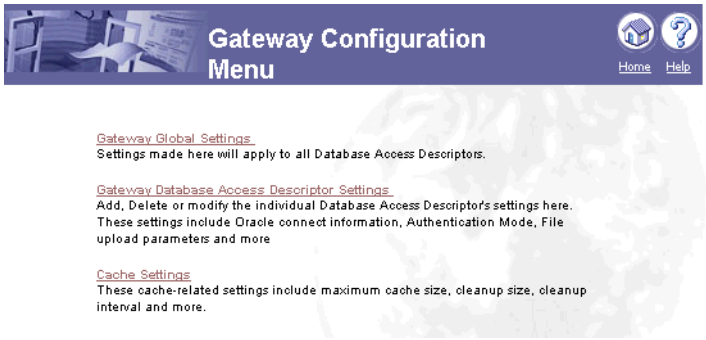
Configuring the PL/SQL Gateway

The PL/SQL Gateway provides a Web page for configuring Database Access Descriptors (DADs). A DAD is a set of values that specify how the PL/SQL Gateway connects to a database server to fulfill an HTTP request.

4.1 Accessing the Gateway Configuration Menu

Access the Gateway Configuration Menu at:

```
http://<hostname>:<port>/pls/admin_/gateway.htm
```



4.1.1 Accessing the Gateway Configuration Menu from Oracle Portal

You can also access the Gateway Configuration Menu through Oracle Portal. To do so:

1. From the Oracle Portal home page, click the Administer tab. By default, this is where the Services portlet is located.

2. In the Services portlet, click Listener Gateway Settings.

4.2 Accessing the Global Settings

From the Gateway Configuration Menu page, access the Global Settings page by clicking the **Gateway Global Settings** link. Or access the Global Settings page directly at:

```
http://<hostname>:<port>/pls/admin_/globalsettings.htm
```

The Global Settings page is shown below.

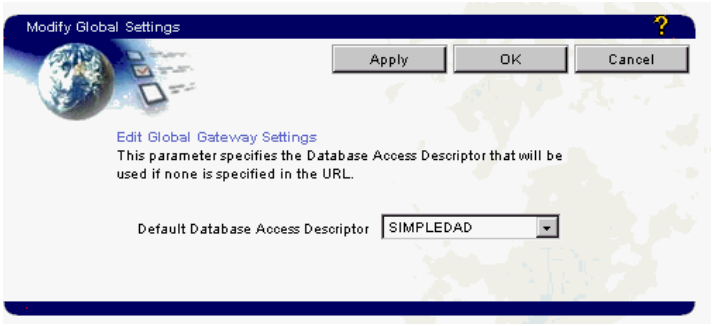


Table 4–1 Global Setting through the Gateway Configuration Page

Global Setting	Description
Default Database Access Descriptor (DAD)	Specifies a path that points to the default DAD. If the end user enters a URL without specifying the DAD name, the home page for the default DAD is displayed. **Default** = none Change the DAD name by typing a new one in this field.

4.2.1 Global Settings Accessible through the Configuration File

Access the following settings through the wdbsvr.app configuration file (not through the Gateway Configuration page). This configuration file describes settings for the PL/SQL Gateway module. The location of your Oracle9*i* Application Server installation is*<ORACLE_HOME>*. For UNIX, the configuration file is located at:

<ORACLE_HOME>/Apache/modplsql/cfg/wdbsvr.app

For NT, it is located at:

<*ORACLE_HOME*>\Apache\modplsql\cfg\wdbsvr.app

Table 4–2 Global Settings in the Configuration File

Global Settings	Description
Administrators (administrators)	Specifies who can view the admin pages. By default, this is set to ALL. Change this to a comma separated list of users to enforce security on the admin pages. For example *scott, mike* where *scott* and *mike* are local database user names. Or, *scott, mike@orcl* where orcl is a connect string for a remote database. Refer to "Protecting the Administration pages" on page 2-5 for more information.
Admin Path (adminPath)	Specifies the URL path element that identifies an admin page. Usually this is left unchanged as */admin_/*.
Admin DAD (admindad)	Specifies the DAD to authenticate the user who can view the admin pages. Refer to "Protecting the Administration pages" on page 2-5 for more information.

Table 4–2 Global Settings in the Configuration File

Global Settings	Description
Error Reporting Mode (error_style)	Specifies the Error Reporting Mode for errors. This parameter can be specified at the DAD level or at the Global level and must be changed by manually editing the file. This parameter accepts the following values: error_style = WebServer (default for mod_plsql) PL/SQL Gateway indicates to the Web Server what HTTP error was encountered. The Web Server then generates the error page. This can be used with Apache's Error Document directive to produce customized error messages. error_stye = Gateway The PL/SQL Gateway generates the error pages, usually a short message indicating the PL/SQL error that was encountered (e.g. "scott.foo PROCEDURE NOT FOUND"). error_style = GatewayDebug The PL/SQL Gateway generates information specified by the "Gateway" setting and also produces server configuration information. This mode is for debugging purposes only. Do not use in a production system since displaying internal server variables could produce a security risk. **Note:** If the PL/SQL Application generates its own error page, then the error_style setting is ignored and the page generated by the application is displayed

4.3 Accessing Database Access Descriptor settings

You can access the Database Access Descriptor Settings page from the **Gateway Database Access Descriptor Settings** link on the Gateway Configuration Menu page, or directly at:

```
http://<hostname>:<port>/pls/admin_/dadentries.htm
```

A section of the Database Access Descriptor Settings page is shown below.

Table 4–3 DAD Settings through the Gateway Configuration Page

DAD Settings	Description
Database Access Descriptor Name	Displays the name for this DAD. The name is set at installation or during the creation of new web sites. Change the name by typing a new one in this field.
Oracle User Name	Displays the Oracle database account user name. The user name is typically set at installation or during the creation of new web sites. Change it by typing a new name in this field.
Oracle Password	Displays the Oracle database account password. The password is typically set at installation, but you can change it by typing a new password in this field. **Note:** The **Oracle User Name** and **Password** are the default user name and password for logging in to a Web site or page. If you leave the **Oracle User Name** and **Oracle Password** fields blank**,** the user is prompted to enter a user name and password when first logging in.

Table 4–3 DAD Settings through the Gateway Configuration Page

DAD Settings	Description
Oracle Connect String	Specify in the format of HOST:PORT:SID if the database is remote. You can leave this field blank if the database is local. For example, if your SQL*Net Listener for database sid ORCL is running on port 1521 of machine mydb.us.oracle.com, you can specify the Oracle Connect String to be "mydb.us.oracle.com:1521:ORCL". The previous mode of specifying a Net8 TNS Alias is also still allowed.
Authentication Mode	This parameter can be set to one of the following values: ■ Basic - authentication is performed using Basic HTTP Authentication. Most applications use Basic Authentication. ■ Global OWA - authorization id performed in the OWA package schema. ■ Custom OWA - authorization is performed using packages and procedures in the user's schema, or if not found, in the OWA package schema ■ PerPackage - authentication is performed using packages and procedures in the user's schema Single Sign-On - authentication is performed using the Oracle Single Sign-On feature of the Login Server. Use this mode only if your application is set up to work with the Login Server.
Session Cookie Name	Enter a session cookie name only for Oracle Portal 3.X installations that participate in a distributed environment.
Create a Stateful Session?	Choose **Yes** to preserve the database package/session state for each database request. Choose **No** to reset it after each request. For the PL/SQL Gateway, this parameter must be set to **No**.
Keep Database Connection Open Between Requests?	Choose whether, after processing one URL request, the database connection stays open to process future requests. Choose **Yes** for maximum performance. Choose **No** for debugging purposes. The PL/SQL Gateway cleanup thread cleans up database sessions that have not been used for 15 minutes. Refer to the "Controlling Database Processes for Each mod_plsql Request" on page B-3, for information regarding database sessions.

Table 4–3 DAD Settings through the Gateway Configuration Page

DAD Settings	Description
Default (Home) Page	Enter the PL/SQL procedure that is invoked when one is not specified as part of the URL. For example, if you specify a default home page of `myapp.home` and an end user enters this URL in a browser: http://myapp.myserver.com:2000/pls/myapp/ It automatically updates the URL to: http://myapp.myserver.com:2000/pls/myapp/myapp.home
Document Table	Enter the name of the database table where files uploaded through a web browser will be stored. Refer to "File Upload and Download" on page 1-12 for more information.
Document Access Path	Enter a path in the URL installation that is used to indicate a document is being referenced. In the following URL, for example: http://myapp.myserver.com:2000/pls/my_site/docs/folder1/presentation.htm `docs` is the document access path.
Document Access Procedure	Enter the procedure for uploading and downloading documents.
Extensions to be Uploaded as LONGRAW	Specify extensions for files to be uploaded as LONGRAW.
Path Alias	To be used by PL/SQL applications for path aliasing. **WebDB 2.X Note:** Leave this field blank if the DAD is for an existing WebDB 2.x Web site. Refer to "Path Aliasing (Direct Access URLs)" on page 1-21 for more information.
Path Alias Procedure	To be used by PL/SQL applications for path aliasing. **WebDB 2.X Note:** Leave this field blank if the DAD is for an existing WebDB 2.x Web site. Refer to "Path Aliasing (Direct Access URLs)" on page 1-21 for more information

4.3.1 DAD Settings Accessible through the Configuration File

Access the following settings through the wdbsvr.app configuration file (not through the Gateway Configuration page). This configuration file describes settings for the PL/SQL Gateway module. The location of your Oracle9*i* Application Server installation is *<ORACLE_HOME>*. For UNIX, the configuration file is located at:

<ORACLE_HOME>/Apache/modplsql/cfg/wdbsvr.app

For NT, it is located at:

<ORACLE_HOME>\Apache\modplsql\cfg\wdbsvr.app

.

Table 4–4 DAD Settings in the Configuration File

DAD Settings	Description
Debug (debugModules	To log messages to Apache logs, ensure that the Apache logging level (LogLevel parameter in httpd.conf) is set to the same level (or higher) as that of debugModules. The error levels mirror that of Apache. Messages with a severity level of Emergency through Info are logged to the Apache error log file $IAS_ HOME/Apache/Apache/logs/*error*. ■ Emerg ■ Alert ■ Crit ■ Error ■ Warn (Default) Logs critical warning/errors to the Apache error logs. ■ Notice ■ Info ■ Debug (all) Similar to the older debugModules=all. Logs information to mod_plsql specific log files created under $IAS_HOME/Apache/modplsql/log. Refer to "Logging" on page B-3 for more information. Note: The following warnings are informational only and relate to performance overhead incurred. Refer to "Overhead Problems" on page B-8 for more information. ■ *Procedure had to be described for execution* - overhead incurred because mod_plsql had to describe the procedure before executing it. ■ *Procedure is using older style flexible parameter passing style* - overhead incurred because less efficient 4-parameter flexible parameter passing is being used.
Environment List (cgi_env_list)	Specifies overriding and/or new CGI environment variables. This is a comma separated list of name and value pairs of environment variables to be added or overridden. Refer to "Adding and Overiding CGI Environment Variables" on page 1-23.

Table 4–4 DAD Settings in the Configuration File

DAD Settings	Description
Error Reporting Mode (error_style)	Specifies the Error Reporting Mode for errors. This parameter can be specified at the DAD level or at the Global level and must be changed by manually editing the file. This parameter accepts the following values: `error_style` = WebServer (default for mod_plsql) PL/SQL Gateway indicates to the Web Server what HTTP error was encountered. The Web Server then generates the error page. This can be used with Apache's ErrorDocument directive to produce customized error messages. `error_stye` = Gateway The PL/SQL Gateway generates the error pages, usually a short message indicating the PL/SQL error that was encountered (e.g. "scott.foo PROCEDURE NOT FOUND"). `error_style` = GatewayDebug The PL/SQL Gateway generates information specified by the "Gateway" setting and also produces server configuration information. This mode is for debugging purposes only. Do not use in a production system since displaying internal server variables could produce a security risk. **Note:** If the PL/SQL Application generates its own error page, then the error_style setting is ignored and the page generated by the application is displayed
Exclusion List (exclusion_list)	Specifies the procedures/packages/schema names which are forbidden to be directly executed from a browser. This is a comma separated list in which each string in the list is case insensitive and can accept wildcards. If this parameter is not specified, the default procedures are used. Refer to "Protecting the PL/SQL Procedures Granted to PUBLIC" on page 2-4 for examples.

Table 4–4 DAD Settings in the Configuration File

DAD Settings	Description

National Language Support (nls_lang)	Specifies the NLS_LANG for this DAD. This parameter overrides the NLS_LANG environment variable. When this parameter is set, the PL/SQL Gateway uses the specified NLS_LANG to connect to the database. Once connected, an alter session is issued to switch to the specified language and territory.
	Furthermore, all data transferred to the PL/SQL Gateway is in the specified charset. For example,
	`nls_lang = JAPANESE_JAPAN.JA16SJIS`
	With this NLS_LANG setting, the PL/SQL Gateway issues the following statement upon connecting to the database:
	`alter session set nls_language=JAPANESE nls_territory=JAPAN`
	All content transferred from the database will be in JA16SJIS charset.

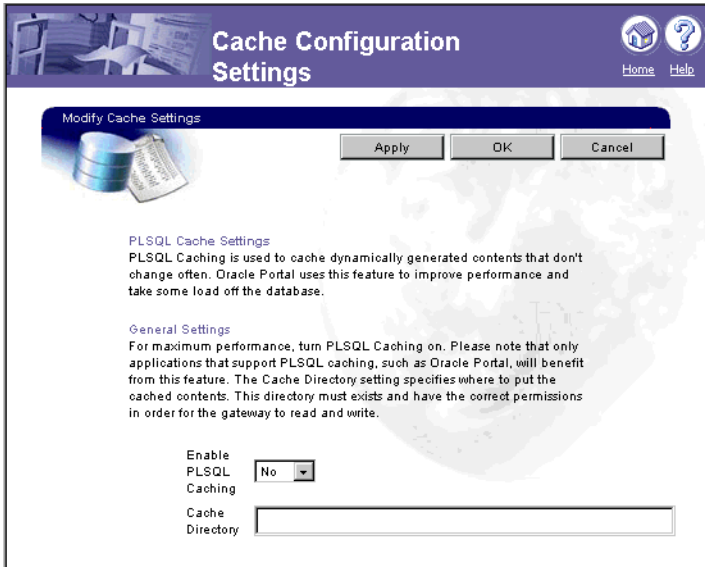
Response Array Size (response_array_size)	Specifies the number of rows of content to fetch from the database for each trip. This performance tuning parameter can affect the response time of each request. For large generated pages, setting this parameter higher can decrease the number of trips to the database to get the content. However, the memory usage increases. The minimum is 28 rows. The default is 128 rows. For Japanese, Chinese or multi-byte character set languages, set this parameter to 256.

4.4 Accessing the Cache settings

Refer to the "Caching" chapter on page 3-1 for more information about caching. Access the Cache Settings page from the **Cache Settings** link on the Gateway Configuration Menu page, or directly at:

`http://<hostname>:<port>/pls/admin_/cache.htm`

This page is split into two subsections for the two types of caching: PL/SQL caching and Session Cookie caching. A section of the Cache Settings page is shown below.



4.4.1 PL/SQL Caching

Table 4–5 PL/SQL Cache Settings

PL/SQL Cache Settings	Description
Enable PL/SQL Caching	**Yes** enables PL/SQL caching for maximum performance. Choose **No** if there are problems relating to this feature.
Cache Directory	Enter a directory that PL/SQL caching uses to store cached content files. **Note:** Ensure that this directory exists and has permissions that allow the PL/SQL Gateway to read and write files to and from it.

Table 4–5 PL/SQL Cache Settings

PL/SQL Cache Settings	Description
Total Cache Size (in bytes)	Specifies the total amount of disk space PL/SQL caching can use. **Note:** This is setting is not a hard limit. The cache may exceed this limit temporarily.
Maximum Cacheable File Size (in bytes)	Specifies the maximum size for all cached files. Any dynamically generated content that exceeds this limit is not cached.
Total Cleanup Limit Size (in bytes)	Specifies the total size of the cache to maintain after the cleanup has occurred. This ensures that frequently accessed contents stay in the cache after cleanup is over.
Cleanup Interval (in seconds)	Specifies the number of seconds the cleanup takes. A high number improves performance, but total cache size may be exceeded. A low number decreases performance, but total cache size is not exceeded.

4.4.2 Session Cookie Caching (Portal only)

Table 4–6 Session Cookie Cache Settings

Session Cookie Cache Settings	Description
Enable Session Caching	Choose **Yes** to enable session cookie caching for maximum performance. Choose **No** if there are problems relating to this feature.
Cache Directory	Enter a directory that session cookie caching uses to store cached content files. **Note**: Ensure that this directory exists and has permissions that allow the PL/SQL Gateway to read and write files to and from it.

Table 4–6 Session Cookie Cache Settings

Session Cookie Cache Settings	Description
Total Cache Size (in bytes)	Specifies the total amount of disk space session cookie caching can use. **Note:** This is setting is not a hard limit. The cache may exceed this limit temporarily.
Total Cleanup Limit Size (in bytes)	Specifies the total size of the cache to maintain after the cleanup has occurred. This ensures that frequently accessed contents stay in the cache after cleanup is over.
Cleanup Interval (in seconds)	Specifies the number of seconds the cleanup takes. A high number improves performance, but total cache size may be exceeded. A low number decreases performance, but total cache size is not exceeded.

.

5

Installing the PL/SQL Gateway

5.1 System Requirements

The following are the recommended and minimum requirements for installing and running the PL/SQL Gateway:

Operating Systems

- Windows NT 4.0 with Service Pack 3 or above
- Solaris 2.6 and above
- HP-UX 11.0
- IBM AIX 4.3.2/4.3.3
- Compaq Tru64 4.0d
- Intel Solaris 2.7

Oracle Database

- Oracle8*i* (Release 8.1.6 or 8.1.7)

Note: The PL/SQL Gateway requires the Oracle 8.1.7 client libraries to be installed in the same Oracle Home as the PL/SQL Gateway. If these libraries are installed, you can still run the PL/SQL Gateway against remote Oracle 8.0.5 or above databases. For example, you can use the PL/SQL Gateway to run PL/SQL procedures installed in a remote 8.0.5 database.

Web Listener

- Oracle HTTP Server (powered by Apache) 1.3.12 for Oracle9*i* Application Server version 1.0.2

Web Browsers

- Netscape 4.0.8 and above
- Microsoft Internet Explorer 4.0.1 with Service Pack 1 and above

5.2 Before you begin

Before you install the PL/SQL Gateway using the Oracle9*i* Application Server v1.0 Oracle Universal Installer, satisfy the following requirements:

- You must have a SYS user password on the database where you plan to load PL/SQL Web Agent packages required by the PL/SQL Gateway.
- The database to which you plan to connect the PL/SQL Gateway must be up and running.
- You must have enough disk space on the machine where you plan to run the Oracle Universal Installer.
- You must have write permissions to the directory where the Oracle Universal Installer is writing its oraInventory data.

5.3 Installation

To begin the Oracle Universal Installer, execute the runInstaller application located on your product CD or stage area. Follow the installation instructions, including choosing a directory where you want to install Oracle9*i* Application Server v1.0.2. This install directory is referred to as <ORACLE_HOME>.

5.4 Installing Required Packages

After installation, manually install additional required packages using the owaload.sql script.

1. Navigate to the directory where the owaload.sql file is located. This directory is `<ORACLE_HOME>/APACHE/modpsql/owa.`
2. Using SQL*Plus, log into the Oracle database as the SYS user.

3. At a SQL prompt, run the following command:

```
@owaload.sql log_file
```

Table 5–1 Installing Required Packages Parameters

Elements	Description
owaload.sql	Installs the PL/SQL Web Toolkit packages into the SYS schema. It also creates public synonyms and makes the packages public so that all users in the database have access to them. Therefore, only one installation per database is needed.
log_file	The installation log file.

5.4.1 Upgrading from Oracle9*i* Application Server or WebDB Listener

If you were previously running Oracle9*i* Application Server or WebDB Listener 2.5 and below:

1. Verify there is no user data (other than the PL/SQL Web Toolkit packages) in the schema.
2. Drop the schema where the old PL/SQL Web Toolkit packages were installed. The new PL/SQL Web Toolkit packages installed in SYS will work.

5.4.2 Upgrading from OAS Installations to Oracle9i Application Server

The PL/SQL Web Toolkit packages shipped with Oracle 8.1.7 and Oracle9*i* Application Server v1.0 are recommended for installation.

If the new PL/SQL Web Toolkit packages have been installed (either automatically or manually) and you were previously running OAS, note that the 8.1.7 install/upgrade or manual install for mod_plsql installs new PL/SQL Web Toolkit packages in the SYS schema. It also recreates PL/SQL Web Toolkit public synonyms to reference these new packages.

However, if you face problems while running the OAS PL/SQL Cartridge, recreate the older public PL/SQL Web Toolkit package synonyms.

To recreate the old public synonyms do the following:

1. From SQL*Plus, connect as SYS

2. Run the following statements. This drops all PL/SQL Web Toolkit public synonyms created during the upgrade process/

```
drop public synonym OWA_CUSTOM;
drop public synonym OWA_GLOBAL;
drop public synonym OWA;
drop public synonym HTF;
drop public synonym HTP;
drop public synonym OWA_COOKIE;
drop public synonym OWA_IMAGE;
drop public synonym OWA_OPT_LOCK;
drop public synonym OWA_PATTERN;
drop public synonym OWA_SEC;
drop public synonym OWA_TEXT;
drop public synonym OWA_UTIL;
drop public synonym OWA_INIT;
drop public synonym OWA_CACHE;
drop public synonym WPG_DOCLOAD;
```

3. Connect to the old PL/SQL Web Toolkit package installation schema (typically OAS_PUBLIC).

4. Run the following statements. This recreates the PL/SQL Web Toolkit public synonyms that were changed during the upgrade process to reference your old PL/SQL Web Toolkit package installation.

```
create public synonym OWA_CUSTOM for OWA_CUSTOM;
create public synonym OWA_GLOBAL for OWA_CUSTOM;
create public synonym OWA for OWA;
create public synonym HTF for HTF;
create public synonym HTP for HTP;
create public synonym OWA_COOKIE for OWA_COOKIE;
create public synonym OWA_IMAGE for OWA_IMAGE;
create public synonym OWA_OPT_LOCK for OWA_OPT_LOCK;
create public synonym OWA_PATTERN for OWA_PATTERN;
create public synonym OWA_SEC for OWA_SEC;
create public synonym OWA_TEXT for OWA_TEXT;
create public synonym OWA_UTIL for OWA_UTIL;
create public synonym OWA_INIT for OWA_CUSTOM;
create public synonym OWA_CACHE for OWA_CACHE;
create public synonym WPG_DOCLOAD for WPG_DOCLOAD;
```

If you have an OAS installation in which the new PL/SQL Web Toolkit packages were never installed and you use Oracle9*i* Application Server, it is recommended that you install the new PL/SQL Web Toolkit packages. If you

continue using the older PL/SQL Web Toolkit packages, in order to use Oracle9*i* Application Server mod_plsql, you must run the following SQL statements

---

Note: These statements are part of the new PL/SQL Web Toolkit package install and are required only if you have never installed the new PL/SQL Web Toolkit packages and choose to continue using the older PL/SQL Web Toolkit packages.

---

1. From SQL*Plus, connect as SYS.
2. Locate the new PL/SQL Web Toolkit packages and install the following packages:

   wpiutl.sql

   wpgdocs.sql

   wpgdocb.sql
3. Grant execute on wpg_docload to public
4. Create public synonym wpg_docload for wpg_docload.

These steps install the required packages needed to run mod_plsql with an older PL/SQL Web Toolkit package installation. In this configuration, you cannot use some of the new features in the PL/SQL Web Toolkit packages.

5.5 Configuring the Oracle HTTP Server Listener

The Oracle9*i* Application Server installation creates configuration files that you can edit, including the following that affect the PL/SQL Gateway.

5.5.1 apachectl file

The apachectl starts and stops the Oracle HTTP Server. For UNIX, it is located at:

```
<ORACLE_HOME>/Apache/Apache/bin/apachectl
```

Inside this file, there are three parameters that affect the PL/SQL Gateway:

Table 5–2 apachectl file Parameters

Parameter	Description
ORACLE_HOME	Oracle Home where the PL/SQL Gateway runs. Default: <ORACLE_HOME>
LD_LIBRARY_PATH (Solaris, Compaq Tru64, Intel Solaris and Intel LINUX) SHLIB_PATH (for HP-UX) LIBPATH (for IBM)	Oracle libraries needed by the PL/SQL Gateway. This must point to an Oracle 8.1.7 installation. Default: <ORACLE_HOME>/lib
WV_GATEWAY_CFG	PL/SQL Gateway configuration file. Default on UNIX: `<ORACLE_HOME>/Apache/modplsql/cfg/wdbsvr.app` Default on Windows NT: `<ORACLE_HOME>\APACHE\modplsql\cfg\wdbsvr.app` On NT, it is an environment variable. To access it, click **Start**->**Settings**->**Control Panel**->**System**. Select the **Environment** tab, then create a System variable called WV_GATEWAY_CFG that points to the configuration file.

Note: To run the PL/SQL Gateway in another Oracle Home:

For UNIX platforms, change both the ORACLE_HOME and LD_LIBRARY_PATH settings. (For HP-UX, change SHLIB_PATH and for IBM, change LIBPATH settings.) For NT, change the PATH environment setting to <ORACLE_HOME>\bin.

5.5.2 httpd.conf file

This configuration file defines the behavior of Oracle HTTP Server (powered by Apache). You can set your port number as well as other server settings. It is located at:

`<ORACLE_HOME>/Apache/Apache/conf/httpd.conf`

5.5.3 plsql.conf file

This configuration file describes settings for the PL/SQL Gateway module. It is located at:

```
<ORACLE_HOME>/Apache/modplsql/cfg/plsql.conf
```

There settings are configurable:

- LoadModule plsql_module <MOD_PATH> - the location of the PL/SQL Gateway module.
  Default on UNIX: `<ORACLE_HOME>/Apache/modplsql/bin/modplsql.so`
  Default on HP: `<ORACLE_HOME>/Apache/modplsql/bin/modplsql.sl`
  Default on Windows NT: `<ORACLE_HOME>\bin\modplsql.dll`
- Location <MOUNT_PATH> - the prefix in the URL for which the PL/SQL Gateway is invoked. Default: /pls

5.5.4 wdbsvr.app file

This main configuration file describes settings for the PL/SQL Gateway module. For UNIX, it is located at:

```
<ORACLE_HOME>/Apache/modplsql/cfg/wdbsvr.app
```

For NT, it is located at:

```
<ORACLE_HOME>\Apache\modplsql\cfg\wdbsvr.app
```

It contains all the DAD information. Most of the settings can be edited by using the PL/SQL Gateway Configuration page, which you can access through your browser as shown "Accessing the PL/SQL Gateway Configuration page" on page 5-9.

5.5.5 Performance Tuning Oracle HTTP Server

This section describes several Oracle HTTP Server *powered by Apache* settings to improve performance.

5.5.5.1 mod_expires

For all static files served by the server (images, static pages, etc.), use mod_expires to cache in the browser. This improves performance by having the browser retrieve the files from the cache, avoiding contacting the server for static files which do not change often.

Mod_expires uses expires-based caching technique. You set the time period that the files are valid for. Do this by using the ExpiresActive, ExpiresDefault, and

ExpiresByType directives. Refer to www.apache.org to search for these directives and their usage.

For example:

To have a directory of static files cached by the browser when they are served, the directives would be:

```
Alias /images/ "/iAS/staic_images/"
<Directory "/iAS/static_images/">
AllowOverride None
Order allow, deny
Allow from all
ExpiresActive On
ExpiresDefault "access plus 8 hours"
</Directory>
```

The browser will cache the material for 8 hours if any of the content in the /iAS/static_images directory is served.

5.5.5.2 KeepAlive Directives

Another family of directives you can use to fine tune performance are the KeepAlive, MaxKeepAliveRequests, and KeepAliveTimeout directives.

These directives tells the server when to destroy a connection between itself and the browser. Since connection establishment is time consuming, this can improve response time if the connection is already established. The default settings are:

Table 5–3 Default Settings

Directive	Default
KeepAlive	On
MaxKeepAliveRequests	100
KeepAliveTimeout	15 seconds

Adjust these settings to achieve better performance. Refer to www.apache.org for more information.

5.5.6 Customizing Error Pages Through the Oracle HTTP Server

For all errors generated by the PL/SQL Gateway, you can customize them to have the same look and feel of your application.

1. Verify the DAD parameter error_style is set to WebServer in your wdbsvr.app file (eg. error_style = WebServer). Refer to "DAD Settings Accessible through the Configuration File" on page 4-8.
2. In your plsql.conf file, between the <Location> and </Location> directives, add the following line:

   ```
   ErrorDocument 500 /error_500.html
   ```

   This tells the Oracle HTTP Server to serve the error_500.html file when it encounters a 500 HTTP status code from the PL/SQL Gateway.
3. For other errors, add similar lines:

   ```
   ErrorDocument 403 /error_403.html
   ```

4. You can also use cgi-scripts to generate dynamic error pages:

   ```
   ErrorDocument 403 /cgi-bin/gen_error.pl
   ```

   This tells the Oracle HTTP Server to execute the perl script called gen_error.pl under the virtual path cgi-bin. For example:

   ```
   <Location /pls>
   SetHandler pls_handler
   Order deny, allow
   Allow from all
   ErrorDocument 500 /error_500.html
   ErrorDocument 403 /cgi-bin/gen_error.pl
   </Location>
   ```

   In this example, any 500 errors from the PL/SQL Gateway will be handled by serving the error_500.html located in the root virtual path. Any 403 errors will be handled by executing the gen_error.pl perl script in the cgi-bin virtual path. For more information about the ErrorDocument directive, refer to Apache's website www.apache.org.

5.6 Accessing the PL/SQL Gateway Configuration page

To access to the PL/SQL Gateway Configuration page, enter the following URL in your Web browser:

```
http://<hostname>:<port>/pls/DAD/<admin_path>/gateway.htm
```

Table 5–4 PL/SQL Gateway Configuration Page Parameters

Parameter	Description
hostname	The machine where the application server is running.
port	The port where the application server is listening. If omitted, port 80 is assumed.
admin_path	The URL path element that identifies an admin page. The default is admin_. For example, if you use the default of admin_, the following URL invokes the PL/SQL Gateway configuration page if the invoking user is listed in the administrators configuration setting: `http://www.myserver.com/pls/admin_/gateway.htm` Configuration settings are protected by the Administrators setting of the configuration file. Refer to "Configuring the PL/SQL Gateway" on page 4-1 for more information.

5.6.1 plsql.conf configuration file

The Oracle HTTP Listener configuration file includes the modplsql configuration file plsql.conf. The contents of plsql.conf are:

```
# Directives added for the PL/SQL Gateway
```

For UNIX:

```
LoadModule plsql_module <ORACLE_HOME>/Apache/modplsql/bin/modplsql.so
```

For HP-UX:

```
LoadModule plsql_module <ORACLE_HOME>/Apache/modplsql/bin/modplsql.sl
```

For NT:

```
LoadModule plsql_module<ORACLE_HOME>\bin\modplsql.dll
```

For all:

```
# Enable handling of all virtual paths beginning with "/pls" by mod-plsql
#
<Location /pls>
  SetHandler pls_handler
```

```
  Order deny,allow
  Allow from all
</Location>
```

5.7 Starting and stopping the Oracle HTTP Server Listener

5.7.1 UNIX

To start the Apache listener, type:

```
<ORACLE_HOME>/Apache/Apache/bin/apachectl start
```

To start the Apache listener with SSL support, type:

```
<ORACLE_HOME>/Apache/Apache/bin/apachectl startssl
```

To stop the Apache listener, type:

```
<ORACLE_HOME>/Apache/Apache/bin/apachectl stop
```

5.7.2 Windows NT

On Windows NT, the Oracle HTTP Server is installed as a service. To start the Oracle HTTP Server with SSL support:

1. Select Start->Settings->Control Panel.
2. Select **Services**.
3. Highlight the **OracleHTTPServer service**.
4. Click **Start**.

To stop the Oracle HTTP Server:

1. Select Start->Settings->Control Panel.
2. Select **Services**.
3. Highlight the **OracleHTTPServer service**.
4. Click **Stop**.

6

PL/SQL Gateway Tutorial

This section provides a step-by-step guide on creating a simple application that displays the contents of a database table as an HTML table. The application invokes a stored procedure that calls functions and procedures defined in the PL/SQL Web Toolkit.

This tutorial assumes the following:

- You have completed the section, "Installing Required Packages" on page 5-2.
- You can log in as the admin user on the server. This is required so that you can add new settings to the server configuration. The database to which you are connecting already has the PL/SQL Web Toolkit installed. Refer to the PL/SQL Web Toolkit Reference Guide for more information.
- You have the SCOTT schema in your Oracle database. The PL/SQL cartridge logs into the database using scott/tiger as the username and password. If you do not have the SCOTTschema, you can use an existing schema on your database, or you can create SCOTT using the CREATE SCHEMA command.

  A schema is a user account containing as a collection of database objects such as tables, views, procedures, and functions. Each object in the schema can access other objects in the same schema.

6.1 Creating and Loading the Stored Procedure into the Database

The stored procedure that the application invokes is current_users (defined below). The procedure retrieves the contents of the all_users table and formats it as an HTML table.

To create the stored procedure, save the text of the procedure in a file called current_users.sql, and then run Oracle Server Manager to read and execute the statements in the file.

1. Type the following lines and save it in a file called current_users.sql. The current_users procedure retrieves the contents of the all_users table and formats it as an HTML table.

```
create or replace procedure current_users
    AS
        ignore boolean;
    BEGIN
        htp.htmlopen;
        htp.headopen;
        htp.title('Current Users');
        htp.headclose;
        htp.bodyopen;
        htp.header(1, 'Current Users');
        ignore := owa_util.tablePrint('all_users');
        htp.bodyclose;
        htp.htmlclose;
    END;
    /
    show errors
```

   This procedure uses functions and procedures from the htp and owa_util packages to generate the HTML page. For example, the htp.htmlopen procedure generates the string

   `<html>`, and `htp.title('Current Users')` generates `<title>Current Users</title>`
   The owa_util.tablePrint function queries the specified database table, and formats the contents as an HTML table.

2. Start up Server Manager in line mode. ORACLE_HOME is the directory that contains the Oracle database files.

```
prompt> $ORACLE_HOME/bin/svrmgrl
```

3. Connect to the database as "scott". The password is "tiger".

```
SVRMGR> connect scott/tiger
```

4. Load the current_users stored procedure from the current_users.sql file. You need to provide the full path to the file if you started up Server Manager from a directory different than the one containing the current_users.sql file.

```
SVRMGR> @  Name of script file: current_users.sql
```

5. Exit Server Manager.

```
SVRMGR> exit
```

6. Configure a DAD to point to the schema where the PL/SQL applications that you want to run with the PL/SQL Gateway are stored. Use the parameters shown in the following table:

Table 6–1 Parameters

Parameter	Value
Database Access Descriptor Name	Scott
Schema	Scott
Oracle User Name	Scott
Oracle Password	Tiger
Oracle Connect String	htmlperf-tcp
Authentication Mode	Basic
Session Cookie Name	
Create a Stateful Session?	No
Enable Connection Pooling	Yes
Maximum Number of Worker Threads	10
Default (Home) Page	Scott.home
Document Table	Scott.wwdoc_document
Document Access Path	docs
Document Access Procedure	Scott.wpg_testdoc.process_download
Extensions to be Uploaded as LONGRAW	*
Path Alias	
Path Alias Procedure	

Note: To require a user to log on to the database containing the application, leave the **Oracle User Name** and **Oracle Password** fields blank.

6.2 Creating an HTML Page to Invoke the Application

To run the current_users procedure, enter the following URL in your browser:

```
http://<host>:<port>/pls/mydad/scott.current_users
```

Or you can invoke the procedure from an HTML page. The following HTML page has a link that calls the URL.

```
<HTML>
<HEAD>
<title>Current Users</title>
</HEAD>

<BODY>
<H1>Current Users</H1>
<p><a href="http://hal.us.oracle.com:9999/simpleApp1/cart1/current_
users">Run
current_users</a>
</BODY>
</HTML>
```

The figure below shows the source page (the page containing the link that invokes the stored procedure), and the page that is generated by the current_users stored procedure.

A

Performance Tuning

This chapter describes several techniques to improve the performance of your PL/SQL application. Go through the worksheet to see if your systems are working to their maximum potential.

Table 6–2 Performance Tuning Worksheet

Parameters and Settings	Recommendations	Refer to:
DAD parameters		
Response Array Size	Default = 128 Japanese, Chinese or Multi-byte Character set languages =Adjust parameter to 256	"Accessing Database Access Descriptor settings" on page 4-4
Keep Database Connection Open Between Requests	Yes=maximum performance No = Debugging	"Accessing Database Access Descriptor settings" on page 4-4 "Controlling Database Processes for Each mod_plsql Request" on page B-3
Oracle HTTP Server *powered by Apache* Settings		
mod_expires	Use to cache in the browser.	"mod_expires" on page 5-7 Refer to www.apache.org for Apache specific information.
KeepAlive directives	Adjust defaults to improve performance. KeepAlive=On MaxKeepAliveRequest=100 KeepAliveTimeout=15	"KeepAlive Directives" on page 5-8 Refer to www.apache.org for Apache specific information.

Table 6–2 Performance Tuning Worksheet

Parameters and Settings	Recommendations	Refer to:
Apache Processes configuration	Adjust defaults to improve performance: MaxRequests=MaxSpareServers MaxRequestsPerchild=HighNumber MinSpareServers=0	"Controlling Database Processes for Each mod_plsql Request" on page B-3. Refer to www.apache.org for Apache specific information.
Caching		
Expires Technique	Best performance (for content that changes predictably)	"Using the Expires Technique" on page 3-7
Validation technique	Good performance (for content that changes unpredictably)	"Validation Technique" on page 3-2
System-level caching	Improves performance by caching one copy for everyone on system	"System- and User-level Caching" on page 3-11

B

Troubleshooting

B.1 OAS Compatibility

B.1.1 Using the mod_plsql without the /pls in the URL

You can use the mod_plsql without the /pls in the URL (similar to the Oracle Application Server [OAS] 4.0.8).

Note: The default installation of mod_plsql is mapped to /pls. The code in the plsql.conf is similar to:

```
[...]
<Location /pls>
  SetHandler pls_handler
  ...
</Location>
[...]
```

There are several methods, outlined below.

- **Rewrite directive (without an external redirect)**
  Since using a redirect or a rewrite engine both cause an external redirect, which adds additional workload, consider using this version of the rewrite directive:

  ```
  [...]
  RewriteEngine on
  RewriteRule ^/myapp/(.*)$ /pls/[DAD]/$1 [PT]
  [...]
  ```

The passthru (PT) flag is a way to post-process the output of RewriteRule directives, but in this case prevents a redirect. Another advantage is that you can hide your URL behind the alias, meaning that the URL does not get rewritten in the browser location window. Therefore this method is the preferred way to set alias names for your application URL.

For more information on rewrites, refer to the documentation at http://www.apache.org/docs/mod/mod_rewrite.html.

- **Redirect the request**
  The PL/SQL Gateway does not map to a filesystem but as a module handler it cannot use an alias or similar mod_alias directives. You can, however, redirect the request. The syntax in the plsql.conf is similar to:

  ```
  Redirect /myapp http://[hostname]:[port]/pls/[DAD]
  ```

  This redirects all requests http://[hostname]:[port]/myapp/... to http://[hostname]:[port]/pls/[DAD]/...

  For more information on rewrites, refer to the documentation at http://www.apache.org for more information on the redirect directive.

- **Use the Rewrite Engine**
  Refer to http://www.apache.org/docs/mod/mod_rewrite.html. The commands offered by this complex module allow rule-based URL manipulations dynamically. The sample rewrites any URL /myapp/... to /pls/[DAD]/ and forces a redirect. The syntax for using the rewrite engine is similar to:

  ```
  [...]
  RewriteEngine on
  RewriteRule ^/myapp/(.*)$ /pls/[DAD]/$1 [R,L]
  [...]
  ```

B.1.2 Using Flexible Parameters and the Exclamation Mark

The Flexible Parameter passing mode in Oracle9*i* Application Server expects the PL/SQL procedure to have the exclamation mark before the procedure name. Due to performance implications of the auto-detect method used in OAS, the exclamation mark is now required for flexible parameter passing in Oracle9*i* Application Server.

In OAS, each procedure is described completely before being executed. The Procedure Describe call figures out the signature of the procedure and requires a round-trip to the database. The PL/SQL Gateway in Oracle9*i* Application Server

avoids this round trip by having end-users explicitly indicate the flexible parameter passing convention by adding the exclamation mark before the procedure. Refer to "Flexible Parameter Passing" on page 1-9 for more information.

B.2 Logging

To turn on logging for mod_plsql, do the following:

1. Stop the Apache Listener.

   Note: When sending log files to Oracle support, start with a clean set of log files to expedite the process. After Step 1., either archive or delete the older logs.

2. Edit the file `<ORACLE_HOME>/Apache/modplsql/cfg/wdbsvr.app`.
3. Change the line containing `debugModules` to `debugModules=all`.
4. Edit the Apache configuration file httpd.conf to set the log level to Info.
5. Restart your listener.
6. When you get an error, stop the Apache listener. The mod_plsql debug log files can be found in `<ORACLE_HOME>/Apache/modplsql/log` directory. Additionally, look at the following log files for errors:

   ```
   <ORACLE_HOME>/Apache/Apache/logs/httpd_access_log
   <ORACLE_HOME>/Apache/Apache/logs/httpd_error_log
   <ORACLE_HOME>/Apache/Jserv/logs/jserv.log
   <ORACLE_HOME>/network/admin/sqlnet.log
   <ORACLE_HOME>/admin/<database>/bdump/alert
   ```

7. When you are finished debugging, revert the changes since enabling logging degrades your web server's performance. Revert the change in the wdbsvr.app file by typing a semi-colon before the line to comment it out.
8. Reset the log level to Warn in the httpd.conf file.

B.3 Controlling Database Processes for Each mod_plsql Request

The database connection pool in mod_plsql is not shared across Apache processes (each process maintains its own pool). The total number of database connections pooled in Apache mod_plsql is directly related to the number of Apache processes

that are spawned and the number of DAD's used to access different PL/SQL applications. The mod_plsql pools one database session per DAD per Apache process. So, the maximum number of database sessions that will be pooled by mod_plsql will be (NumberOfApacheProcesses*NumberOfDADs).

On Windows NT, where Apache is multi-threaded, all threads share the same database connection pool. The maximum number of database sessions that will be pooled by mod_plsql is (MaximumNumberOfApacheThreadsConcurrentlyActive ForEachDAD). This is the ideal case scenario where every thread can take advantage of a database session created by another thread.

On platforms where Apache is not multi-threaded, it is important that the following parameters be tuned.

- Tune the Apache process configuration so that processes are not started or shutdown heavily (each process takes down its connection pool with it, and a new process will need to replenish its pool). This tuning is governed by the load on the Web Server. Set the following: Set MaxClients = MaxSpareServers, and MaxRequestsPerchild=HighNumber and MinSpareServers=0. This configuration ensures that Apache processes are very rarely shutdown and the overhead of creating an Apache process or new database connection is greatly reduced.
- Set up the maximum number of database sessions according to the maximum number of Apache processes expected.
- For heavily loaded sites, turn KeepAlive off (Apache's recommendation). Otherwise, one Apache process gets blocked for the duration of "KeepAliveTimeOut" setup in httpd.conf.

If your database is unable to handle the load, do one of the following:

- Set up a new Listener for handling PL/SQL requests. The main Apache Listener can redirect all PL/SQL requests to the new Listener. For the new Listener, you can set up the Apache processes parameter to a low number (since it will not be doing any other request except PL/SQL). This will keep the database session number to a minimum.
- Consider using Oracle Multi-Threaded Server (MTS). This should reduce the load on the database.
- Consider using the Calypso/iCache/mod_plsql OWA based-caching approaches.

The mod_plsql in Oracle9*i* Application Server v1.0.2 has a cleanup thread which periodically cleans up database sessions which are inactive for more than 15

minutes. So, you need not be concerned about database related resources since mod_plsql does the cleanup automatically at idle time.

B.4 Connection Pooling in mod_plsql

The connection pooling logic in mod_plsql is best explained with an example. Here are the details for a typical scenario:

1. The Oracle9i Application Server listener is started up (There are no database connections in the connection pool maintained by mod_plsql).
2. A browser makes a mod_plsql request (R1) for DAD (D1).
3. One of the Apache processes (Process P1) starts servicing the request R1
4. mod_plsql in Process P1 checks its connection pool and finds that there are no database connections in its pool.
5. Based on the information in DAD D1, mod_plsql in Process P1 opens a new database connection, services the PL/SQL request, and adds the database connection to its pool
6. From this point on, all subsequent requests to Process P1 for DAD D1 can now make use of the database connection pooled by mod_plsql.
7. If a request for DAD D1 gets picked up by another process (say Process P2), then mod_plsql in Process P2 opens its own database connection, services the request and adds the database connection to its pool.
8. From this point on, all subsequent requests to Process P2 for DAD D1 can now make use of the database connection pooled by mod_plsql.
9. Now, assume that a request R2 is made for DAD D2 and this request gets routed to Process P1.
10. mod_plsql in Process P1 does not have any database connections pooled for DAD D2, and a new database session is created for DAD D2 and pooled after servicing the request. Process P1 now has two database connections pooled (one for DAD D1 and another for DAD D2)

The main things to note in this model is that

- Each Apache process serves all kinds of requests (static files requests/Servlet requests/mod_plsql requests) and there is no control on which Apache process will service the next request.

- One Apache process cannot use / share the connection pool created by another process.
- Each Apache process pools at most one database connection for each DAD
- Windows NT has a threaded model for Apache, and the connection pool is shared between all the threads. In this model, the maximum number of database connections created by mod_plsql for each DAD will never exceed the maximum number of simultaneous requests for the DAD.

B.4.1 Install a Separate Apache Listener for mod_plsql Requests

On platforms where Apache is process based (e.g. Solaris), each process serves all kinds of HTTP requests (servlets / PLSQL / static file etc). In a single Oracle9*i* Application Server listener setup, each Apache process maintains its own connection pool to the database. The maximum number of database sessions is governed by the setting in httpd.conf for StartServers / MinSpareServers / MaxSpareServers and the load on the system. This architecture does not allow you to tune the number of database sessions based on the number of mod_plsql requests.

To tune the number of database sessions based on the number of mod_plsql requests, install a separate Apache listener for just mod_plsql requests.

For example: Assume your main Oracle9*i* Application Server Listener is running on port 7777 of mylsnr1.us.oracle.com. You can install another Oracle9*i* Application Server Listener on port 8888 on mylsnr2.us.oracle.com. Then, redirect all mod_plsql requests made to mylsnr1.us.oracle.com:777 to the second listener on mylsnr2.us.oracle.com:8888. This is achieved by doing the following:

1. For the Oracle9*i* Application Server Listener running on Port 7777, edit $IAS_HOME / Apache / modplsql / cfg / plsql.conf in the following manner:
   - Disable servicing of PL / SQL requests from this listener by commenting out the lines between the two Location parameters.

     ```
     <Location /pls>
     ...
     </Location>
     ```

     Note: Comment out lines by putting a # in front of the line.

   - Comment out the line:

```
LoadModule plsql_module...
```

- Configure this listener to forward all mod_plsql requests to the second listener by adding the following line:

  ```
  ProxyPass /pls/ http://my.lsnr2.us.oracle.com:8888/pls/
  ```

2. For the Oracle9*i* Application Server Listener running on Port 8888, configure each DAD to override the default CGI environment variables HOST, SERVER_NAME, SERVER_PORT so that PL/SQL procedures which construct URLs can do so without any problems. To do this, edit $IAS_HOME/Apache/modplsql/cfg/wdbsvr.app and add the following line for each DAD:

   ```
   cgi_env_list=SERVER_NAME=mylsnr1.us.oracle.com,SERVER_
   PORT=7777,HOST=mylsnr1.us
   oracle.com:7777
   ```

In this setup, Apache processes on the main listener lsnr1.us.oracle.com can be configured based on the load to the Oracle9*i* Application Server Listener. Then the Apache processes on the second listener can be fine-tuned based on the mod_plsql requests being made.

B.5 Debugging

The following steps can help you isolate your problem. Verify you can do each step before proceeding to the next one.

1. If you are getting Login failures, confirm that your DAD configuration in <ORACLE_HOME>\Apache\modplsql\cfg\wdbsvr.app has the proper user name, password and connect string. Use SQL*Plus to verify that you have connectivity to the database.
2. If you have multiple Oracle homes, verify that the New8 alias is present in the correct tnsnames.ora file. Synchronize all versions of tnsnames.ora to have the same content. Merge the files instead of directly copying one over the other.
3. If you can logon, try accessing a simple procedure, for example:

   ```
   http://host:port/pls/DAD/htp.p?cbuf=Hello
   ```

4. If you still have problems, contact Oracle Support and provide the log files. Refer to the "Logging" section on page B-3 for instructions on obtaining a log.

B.5.1 Error Code 503 (Service Temporarily Unavailable)

If you are getting error code "503 Service Temporarily Unavailable" when your web server is under some load, determine if the mod_plsql cannot connect to the database because the maximum number of database sessions has been reached.

- Check the maximum number of sessions parameter in your database configuration file. This parameter is called `processes` and is in your database configuration file init$SID.ora. This number must be at least equal to the maximum number of Apache processes configured in httpd.conf (StartServers+MaxSpareServers).
- Verify this is the issue by doing the following.
  1. As soon as you get this error, connect as SYS through SQL*Plus.
  2. Issue "select username from v$session".
  3. If the count of the number of rows is almost the same as what you have in init$SID.ora, then you are having this problem.

B.6 Symbols Displaying Incorrectly with Netscape 6

There is an issue with symbols outside the range of the US-ASCII charset displaying correctly with Netscape 6. If your symbols are being displayed as question marks, implement the following fix:

Enter the following to emulate the various symbols:

Table 6–3 Code to emulate restricted symbols

Restricted Symbols	Code
Trademark symbol (™)	`<SUP><SMALL>TM</SMALL></SUP>`
Copyright symbol (©)	`&copy`
Registration symbol (®)	`&reg`

B.7 Overhead Problems

While executing some of the Portal stored procedures, mod_plsql is incurring a Describe overhead which results in two extra round trips to the database for a successful execution. This has performance implications.

B.7.1 The Describe Overhead

In order to execute PL/SQL procedures, mod_plsql needs to know about the data type of the parameters being passed in. Based on this information, mod_plsql binds each parameter either as an array or as a scalar. One way to know the procedure signature is to describe the procedure before executing it. However, this approach is not efficient because every procedure has to be described before execution (unless the procedure descriptions are cached).

To avoid the "describe" overhead, mod_plsql looks at the number of parameters passed for each parameter name. It uses this information to assume the datatype of each variable. The logic is simply, "if there is a single value being passed, then the parameter is a scalar, otherwise it is an array".

This works for most cases but fails if someone tries to pass a single value for an array parameter. Then the PL/SQL procedure execution fails because it binds an array as a scalar. In such circumstances, mod_plsql issues the Describe call for the procedure and binds each parameter based on the information retrieved from the describe call, reexecutes the procedure and sends back the results. This happens transparently to the procedure, but internally mod_plsql has encountered two extra round trips (one for the failed execute and the other for the "Describe" call).

B.7.2 Avoiding the Describe Overhead

You can avoid performance problems by doing the following:

- Use Flexible Parameter Passsing.
- Always ensure that you pass multiple values for arrays (for single values, you can pass dummy values which will be ignored by your procedure).
- Using the following workaround. The workaround defines a two-parameter style procedure which defaults the unused variables.

  1. Define a scalar equivalent of your procedure which internally calls the original procedure. For example, if the original package is similar to the following, create or replace package testpkg as:

     ```
     TYPE myArrayType is TABLE of VARCHAR2(32767) index by binary_integer;
     procedure arrayproc (arr myArrayType);
     end testpkg;
      /
     ```

  2. If you are making URL calls like "/pls/.../testpkg.arrayproc?arr=1", change the specification to be similar to the following:

```
create or replace package testpkg as
  TYPE myArrayType is TABLE of VARCHAR2(32767) index by binary_integer;
  procedure arrayproc (arr varchar2);
  procedure arrayproc (arr myArrayType);
end testpkg;
/
```

3. Create or replace package body testpkg as:

```
procedure arrayproc (arr varchar2) is
  localArr myArrayType;
begin
  localArr(1) := arr;
  arrayproc (localArr);
end arrayproc;
```

B.7.3 The Flexible Parameter Passing (four parameter) Overhead

A similar round-trip overhead exists if a PL/SQL procedure is using the older style four parameter interface. The PL/SQL Gateway first tries to execute the procedure by using the two parameter interface. If this fails, the PL/SQL Gateway tries the four parameter interface. This implies that all four parameter interface procedures will experience one extra round-trip for execution.

B.7.4 Avoiding the Flexible Parameter Passing Overhead

To avoid this overhead, it is recommended that you write corresponding wrappers which use the two parameter interface and internally call the four parameter interface procedures. Another option is to change the specification of the original procedure to default the parameters which are not passed in the two parameter interface.

B.8 Setting up PL/SQL to Work with WebDB

WebDB users running WebDB version 2.x (2.0, 2.1, 2.2) through the PL/SQL Gateway must perform the following steps.

1. Drop any older OWA packages in OWA_PUBLIC or OAS_PUBLIC.
2. Install the latest OWA packages shipped with the PL/SQL Gateway. To do so, connect to the database as the SYS user and at the command prompt run

   ```
   @owaload.sql log.txt
   ```

This may invalidate some of your existing PL/SQL procedures. You may need to recompile them. Refer to "Installing Required Packages" on page 5-2 for more information

3. Set the following in the DAD configuration for the WebDB 2.x schema in wdbsvr.app configuration file.

   Authentication Mode = Basic

   Document Table = schema.wwv_document

   Extensions to be Uploaded as Long Raw = *

   If you set up your DAD using the **Add for WebDB 2.x configuration** page (http://<hostname>:<port>/pls/admin_/dadentries.htm), these settings are automatically set.

4. Connect to the database as the owner of the site and run wwvdocs.sql and wwvdocb.plb to enable WebDB 2.x sites. These files are located in the same directory as the owaload.sql file. Refer to "Installing Required Packages" on page 5-2 for more information.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W