Oracle9*i* Application Server

Using the PL/SQL Gateway

Release 1.0.2.1

January 2001

Part No. A87562-01

ORACLE®

Oracle 9*i* Application Server

Using the PL/SQL Gateway

Release 1.0.2.1

Part No. A87562-01

Primary Author: Cheryl Smith

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

Contents

2 Securing applications through the PL/SQL Gateway

3 Installing the PL/SQL Gateway

4 Configuring the PL/SQL Gateway

5 Using the PL/SQL Web Toolkit

Preface

This manual describes how to install, configure, and maintain the PL/SQL Gateway for Oracle9*i* Application Server 1.0.2.1. 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	Explains how to install the PL/SQL Gateway.
Chapter 4	Describes global PL/SQL Gateway settings, and individual Database Access Descriptor and Cache settings.
Chapter 5	Describes how to install the PL/SQL Web Toolkit. Before you use the PL/SQL Gateway, you must install the packages in the PL/SQL Web Toolkit in the SYS schema of your Oracle database.
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	Describes special considerations for running Oracle WebDB versions 2.0, 2.1, and 2.2 with the PL/SQL Gateway.
Appendix B	Describes common problems and solutions.

Related Documents

For more information, see the following manuals:

- *Oracle9i Application Server, Release 1.0.2 - Migrating from Oracle Application Server A83709-03*

- *Oracle9i Application Server, Release 1.0.2 - Overview A87353-01*

Send Us Your Comments

Oracle9*i* Application Server

Using the PL/SQL Gateway

Release 1.0.2.1

Part No. A87562-01

Oracle Corporation welcomes your comments on the 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, and telephone number.

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

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 3.0.

1.1 PL/SQL Gateway Configurations

Oracle9*i* Application Server provides two configurations for deploying PL/SQL-based Web applications:

- Oracle9*i* Application Server and mod_plsql support running in stateless mode only. This is the recommended configuration for users developing stateless PL/SQL-based Web applications. In the stateless mode, mod_plsql has a connection pooling mechanism that can keep database sessions open between HTTP requests.

  In a stateless environment, each HTTP request from a client maps to a new database session. Application state is typically maintained in HTTP cookies or database tables. Transaction state cannot span across requests. If a PL/SQL procedure executes successfully, an implicit commit is performed. If it executes with an error, an implicit rollback is performed.

- Oracle9*i* Application Server and mod_ose support running in both stateless and stateful mode. This is the recommended configuration for users developing stateful PL/SQL- and Java-based Web applications. 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.

In a stateful environment, each HTTP request from a client maps to the same database session. Application state is preserved in PL/SQL package variables. A transaction can span across requests because no implicit commits or rollbacks are performed.

1.2 Database Access Descriptors

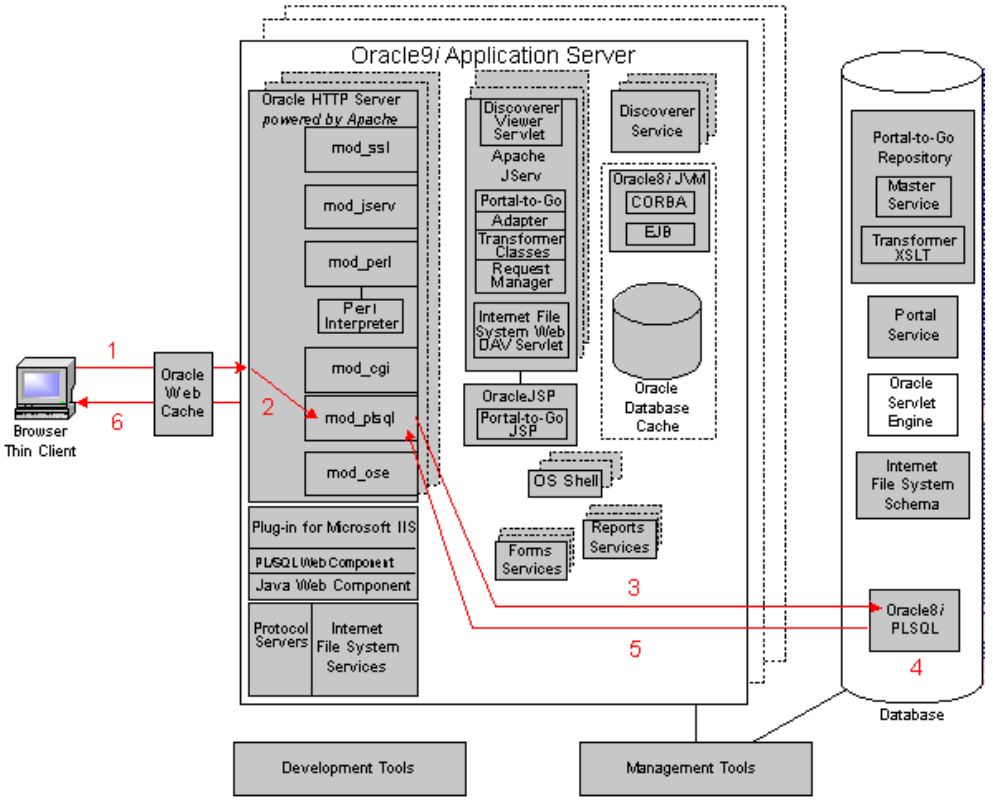
Each PL/SQL Gateway request is associated with a Database Access Descriptor (DAD), a set of configuration values used for database access. A DAD specifies information such as:

- 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, see "Authenticating Users" on page 2-1.

1.3 Processing Client Requests

The following occurs when a server receives a request:



1. The Oracle HTTP Server receives a PL/SQL Server Page request, through Oracle Web Cache, from a client browser.
2. The Oracle HTTP Server routes the request to mod_plsql.
3. The request is forwarded by mod_plsql to Oracle8*i* PLSQL. By using the configuration information stored in your DAD, mod_plsql connects to the database, prepares the call parameters, and invokes the PL/SQL procedure in the database. See "Configuring the PL/SQL Gateway" on page 4-1 for more information.
4. The PL/SQL procedure generates an HTML page using data and stored procedures accessed from the database.
5. The response is returned to mod_plsql.

6. The Oracle HTTP Server sends the response, through Oracle Web Cache, 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 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. See "Using the PL/SQL Web Toolkit" on page 5-1 for more information.

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*`]]`

where:

Table 1–1 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 1-8 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.

Table 1–1 Invoking the PL/SQL Gateway Parameters

Parameter	Description
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.
?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 and GET Methods" on page 1-6 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 1-6 for more information.

1.4.1 POST and GET Methods

The POST and GET 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 either method. Each method is as secure as the underlying transport protocol (http or https).

- When you use 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 you use 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.

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 1-1.

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 1-7 for more information.

- **Flexible parameter passing**

  Procedures are prefixed by a ! character. See "Flexible Parameter Passing" on page 1-8 for more information.

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

  See "Large Parameter Passing" on page 1-10 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],
```

where:

Table 1–2 Two Parameter Interface Parameters

Parameter	Description
proc_name (optional)	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 (optional)	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

```
procedure [proc_name] is
         (num_entires IN NUMBER,
```

```
            name_array  IN  [array_type],
            value_array IN  [array_type],
            reserved in [array_type]);
```

where:

Table 1–3 Four Parameter Interface Parameters

Parameter	Description
proc_name (optional)	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 (optional)	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 four 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 1-8), 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 1-17.

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`.

1.7.6 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 1-15 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 document download packages are in the PL/SQL Web Toolkit. See "Installing Required Packages" on page 3-2 for more information.

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 downloaded 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 file 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.

1.8 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 Caching

Caching can improve performance of PL/SQL Web applications. You can cache PL/SQL procedures and Web content in the middle-tier. Subsequent requests for the content may be retrieved from the cache, with or without validation from the database, thereby decreasing the database workload. When enabled, caching increases the scalability of your Web application.

There are a number of cache mechanisms in the HTTP protocol suite. The HTTP protocol consists of Requests and Responses. A user agent, for example a Web browser, can supply metadata in the Request Headers.

Content providers such as PL/SQL procedures can supply the cache-controlling metadata using one or more HTTP Response Headers. In subsequent HTTP requests, this metadata is supplied by the user agent so that the content provider can determine the validity of the user agent's cache entry.

In cases such as the **Expires** Response Header, the metadata indicates that the content is valid for a certain period of time. Subsequent requests need not be made until that period of time elapses. The content provider has indicated that it need not be reaccessed for a period of time, although the user agent may still do so.

1.9.1 Validation technique

When a Web page is initially generated, it contains a **Last-Modified** Response Header. This header indicates the date, relative to the server, of the content that was requested. User agents with caching capabilities save this date information along with the content. When subsequent requests are made for the URL of the Web page, the user agent:

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 content provider.

Cache-enabled content providers 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 user agent 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 user agent must replace its cache entry with the new content and new date information.

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 user agent. Content providers 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 date method. Content providers generate the ETag header value as part of the Response Header. The user agent stores this opaque header value along with the content that is steamed back. When

the next request for this content arrives, the user agent passes the **If-Match** header with the opaque value that it stored to the content provider. Because the content provider generated this opaque value, it is able to determine what to send back to the user agent. The rest is exactly like the **Last-Modified** validation method as described above.

1.9.2 Expires technique

If a Web page contains an Expires Response Header, the user agent may use this date value, combined with the Date Response Header, to determine how long the response is valid. It does not need to contact the content provider during this time because the validity criteria have already been established. Therefore, the user agent can directly stream back the cached content for that request.

1.9.3 The PL/SQL Gateway Cache

Using the HTTP protocol as the design basis, the PL/SQL Gateway serves as a user agent and a PL/SQL procedure serves as the content provider. Similar to HTTP, headers and environment variables are the communication mechanism between the user agent and the content provider.

One of the assumptions is that the content being cached is varying and typically secured on a per user basis, although the physical URL being cached might be the same across users. Furthermore, content might be in different languages. These assumptions make this design different from the HTTP/1.1 protocol, which uses only the URL to create a cache key. The PL/SQL Gateway uses the URL in conjunction with the user and language to form the cache key.

There are two levels of caching for each request:

- **User-level caching** is for a specific user that is logged in. The stored cache is unique for that user. Only that user gets to use the cache.
- **System-level caching** is for a group of users that shares the cache.

For example, if no individual user chooses to customize a customizable PL/SQL Web application, then the application's output can be stored in a system-level cache. Therefore, there is only a single cache copy for every user on the system.

However, if one of the users customizes the application, then a new user-level cache is stored for that user only. All other users still use the system level cache. This is explained in more detail in "System- and User-level Caching" on page 1-25.

1.9.3.1 owa_cache package

The owa_cache package contains functions and 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 (see "Installing Required Packages" on page 3-2 for more information). See "owa_cache" on page 5-6 for a complete specification of the owa_cache package.

These are the primary functions to call:

- `owa_cache.set_cache(p_etag IN varchar2, p_level IN varchar2)`

  This function sets up the headers for the validation model of caching. The p_etag parameter is the string that tags the generated content. The p_level parameter is the caching level to use.

- `owa_cache.set_expires(p_expires IN number, p_level IN varchar2)`

  This function sets up the headers for the expires model of caching. The p_expires parameter is the number of minutes the generated content is valid. The p_level parameter is the caching level to use.

- `owa_cache.set_not_modified`

  This function is only valid for the validation model. It sets up the headers to notify the gateway to use the cached content.

- `owa_cache.get_level`

  This function is only valid for the validation model. It gets the caching level, "USER" or "SYSTEM".

- `owa_cache.get_etag`

  This function is only valid for the validation model. It gets the tag associated with the cached content.

1.9.3.2 Validation Model

This model is similar to the HTTP **ETag** caching technique. The PL/SQL Gateway always asks the PL/SQL procedure whether the content has changed or not.

Assume a PL/SQL procedure is called for the first time through the PL/SQL Gateway. The PL/SQL Gateway executes the procedure and passes the usual Common Gateway Interface (CGI) environment variables. The procedure generates

content to pass back. If the procedure decides that the generated content is cacheable, it calls the owa_cache procedure to set the tag and the cache level:

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

where:

Table 1–4 Validation Model Parameters

Parameter	Description
`set_cache` function	Sets up the headers to notify the PL/SQL Gateway that the content being streamed back can be cached. Then, the PL/SQL Gateway 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).

Next, assume a second request for the same PL/SQL procedure. The PL/SQL Gateway detects that it has a cached content for the request. In this case, it does something special: it passes that same tag and caching level information, which it got last time from executing the same procedure, as part of the CGI environment variables. The procedure then uses these caching CGI environment variables to check if the content has changed. It does so by calling the following `owa_cache` functions:

```
owa_cache.get_etag;
```

```
owa_cache.get_level;
```

These functions get the tag and caching level respectively. Since the PL/SQL procedure generated these the last time, it can do any kind of processing on them to determine whether the content needs to be regenerated or not.

If the content is still the same, the procedure calls the following `owa_cache` procedure:

```
owa_cache.set_not_modified;
```

and generates no content. This tells the PL/SQL Gateway to use its cached content for this request. Therefore, the cached content is directly streamed back to the browser.

On the other hand, if the PL/SQL procedure determines that the content has changed, it generates the new content along with a new tag and caching level. It does not call `owa_cache.set_not_modified` because the PL/SQL Gateway has a stale copy of the content. Instead the PL/SQL Gateway replaces its stale cached copy with the newly generated one and updates the tag and caching level information associated with it.

1.9.3.3 Expires Model

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.

Assume the same scenario described above for the Validation model, except the procedure uses the Expires model for caching. Once it has generated the content, the procedure calls the following owa_cache procedure:

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

where:

Table 1–5 Expires Model Parameters

Parameter	Description
`set_expires` procedure	Sets up the headers to notify the PL/SQL Gateway that Expires model caching is being used. The PL/SQL Gateway 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.

Next, assume the same procedure invoked a second time through the browser. The PL/SQL Gateway detects that it has a cached copy of the content that is expires-based, then checks for its validity by taking the difference between the current time and the time this cache file was created. If this difference is within the validity period, the cached copy is still fresh and served to the browser without any database interaction.

If the difference is not within the validity period, the cached copy is stale. In this case, the PL/SQL Gateway invokes the procedure. The procedure then decides

whether to use expires-based caching again. Alternatively, it can use the validation model caching or no caching at all.

1.9.4 System- and User-level Caching

The 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).
- For user-level content, it passes the string "USER" as the parameter for the caching level.

The difference in the PL/SQL Gateway between system- and user-level caching is how the user information is applied. For system-level caching, user information is not used since the cache can be used by multiple users.

For a user-level cache hit, the user information is a criteria. A user-level cache always overrides a system-level cache. If both a system-level cache and a user-level cache copy exist for a user, the user-level cache is applied.

1.10 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 an alphabetical list of the available CGI Environment Variables:

- AUTHORIZATION
- DAD_NAME
- DOC_ACCESS_PATH
- DOCUMENT_TABLE (see "document_table (document_table_name)" for more information)
- 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 (see "REMOTE_USER CGI Environment Variable" on page 2-4 for more information)
- REQUEST_CHARSET (see "REQUEST_CHARSET CGI environment variable" on page 1-28 for more information)
- 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;
```

where:

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

1.10.1 Overriding 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 1-25), 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.

Access `cgi_env_list` through the PL/SQL Gateway configuration file (`wdbsvr.app`). This 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

where *<ORACLE_HOME>* is the location of your Oracle9i Application Server installation.

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".

1.10.2 NLS

For PL/SQL Gateway mod_plsql, 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.

1.10.2.1 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), this is the character set information derived from the NLS_LANG environment variable.

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

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

1.10.2.2 REQUEST_IANA_CHARSE 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. Whereas the Web server protects documents, virtual paths and so forth., 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 for the PL/SQL application developer, not for everyone using the application. Some procedures in the dbms_% packages allow anyone to obtain

sensitive information and schedule batch jobs through a browser if they are not protected.

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`). This 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`

where *<ORACLE_HOME>* is the location of your Oracle9i Application Server installation.

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.

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`. This 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`

   where *<ORACLE_HOME>* is the location of your Oracle9i Application Server installation.

2. In the WVGATEWAY section, typically 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`. This 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

   where `<ORACLE_HOME>` is the location of your Oracle9i Application Server installation.

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
   ```

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

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

Installing the PL/SQL Gateway

3.1 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

3.2 Before you begin

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

3.3 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>.

3.4 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/modplsql/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
```

where:

Table 3–1 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.

3.4.1 Installing PL/SQL Web Toolkit Packages

Starting with Oracle 8.1.7 and Oracle9*i* Application Server v1.0.1, there are a new set of PL/SQL Web Toolkit packages which have additional functionality.

- In an Oracle 8.1.7 install/upgrade, the new PL/SQL Web Toolkit packages are automatically installed into the SYS schema.
- In Oracle9*i* Application Server v1.0.1, users of mod_plsql are required to manually install these packages into the SYS schema.
- In an 8.1.7 install, the new PL/SQL Web Toolkit packages are located under the <ORACLE_HOME>/rdbms/admin directory.
- In an Oracle9*i* Application Server v1.0.1 install, the new PL/SQL Web Toolkit packages are located under the <ORACLE_HOME>/Apache/modplsql/owa directory.

Note that Oracle Portal 3.0 depends on the new PL/SQL Web Toolkit packages.

3.4.1.1 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.

3.4.1.2 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.

3.5 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.

3.5.1 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 3–2 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

Table 3–2 apachectl file Parameters

Parameter	Description
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..

3.5.2 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

3.5.3 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

3.5.4 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 below.

3.6 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

where:

Table 3–3 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.

Table 3–3 PL/SQL Gateway Configuration Page Parameters

Parameter	Description
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. See "Configuring the PL/SQL Gateway" on page 4-1 for more information.

3.6.1 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 both:

```
# Enable handling of all virtual paths beginning with "/pls" by mod-plsql
#
<Location /pls>
  SetHandler pls_handler
  Order deny,allow
  Allow from all
</Location>
```

3.7 Starting and stopping the Oracle HTTP Server Listener

3.7.1 UNIX

To start the Apache listener, type:

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

To start the Apache listener with SSL support, type:

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

To stop the Apache listener, type:

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

3.7.2 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**.

4

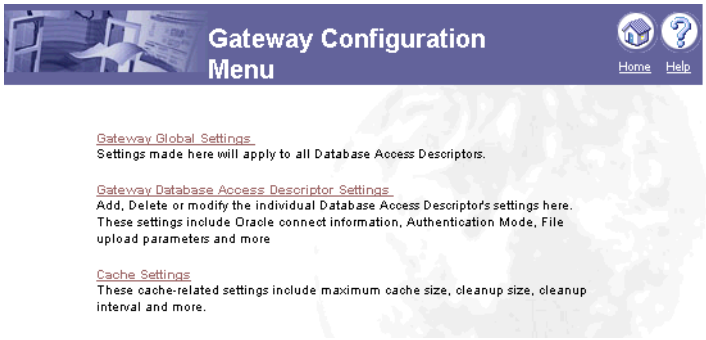
Configuring the PL/SQL Gateway

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 Global Settings

Access the Gateway Configuration Menu at:

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



From this 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.

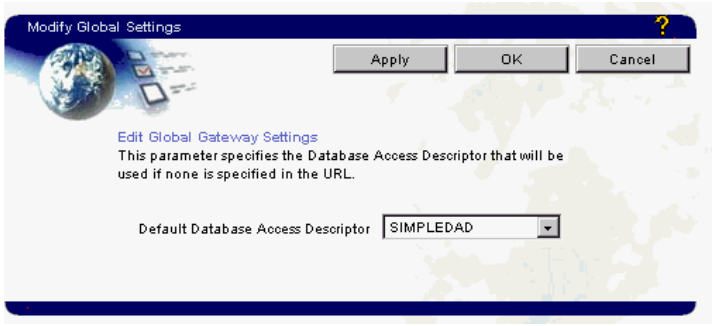


Table 4–1 Global Setting through the Gateway Configuration Page

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.

4.1.1 Global Setting 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. 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`

where `<ORACLE_HOME>` is the location of your Oracle9i Application Server installation.

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 2-6 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 2-6 for more information.

4.2 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.
Oracle Connect String	Enter a Net8 alias if you are using a remote database. Leave this field blank if the database is local.

Table 4–3 DAD Settings through the Gateway Configuration Page

DAD Settings	Description
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. In most configurations, choose **Yes** for maximum performance. The PL/SQL Gateway cleanup thread cleans up database sessions that have not been used for 15 minutes.
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

Table 4–3 DAD Settings through the Gateway Configuration Page

DAD Settings	Description
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 1-11 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 1-18 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 1-18 for more information

4.2.1 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. 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

where *<ORACLE_HOME>* is the location of your Oracle9i Application Server installation.

.

Table 4–4 DAD Settings in the Configuration File

DAD Settings	Description
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 "Overriding CGI Environment Variables" on page 1-27.
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.
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 fetch the content. However, the memory usage increases. The minimum is 28 rows. The default is 128 rows.

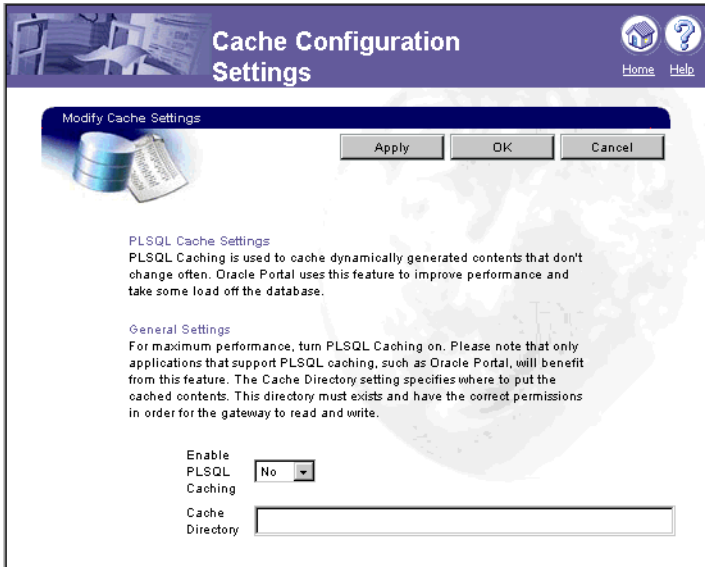
4.3 Cache settings

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.



4.3.1 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.

4.3.2 Session Cookie Caching

Table 4–6 Session Cookie Cache Settings

Session Cookie Cache Settings	Description
Enable PL/SQL 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

Using the PL/SQL Web Toolkit

Before you use the PL/SQL Gateway, you must install the packages in the PL/SQL Web Toolkit in the SYS schema in your Oracle database.

Note: If you installed the Oracle 8.17 database, the packages are by default installed in this schema.

Public synonyms enable users to execute the objects in the common schema. Then users execute the objects in the common schema with their own privileges, rather than with the privileges of the common schema.

If multiple instances of the PL/SQL Web Toolkit are installed in the database, it is recommended that you drop earlier packages from the individual schemas.

5.1 PL/SQL Web Toolkit Installation

If you did not install the PL/SQL Web Toolkit when you installed the PL/SQL Gateway, install it using the owaload .sql installation script. See "Installing Required Packages" on page 3-2 for more information.

5.2 Packages in the Toolkit

The PL/SQL Web Toolkit contains the following packages:

Package	Description
htf and htp	The htp (hypertext procedures) package contains procedures that generate HTML tags. For instance, the htp.anchor procedure generates the HTML anchor tag, <A>. The htf (hypertext functions) package contains the function version of the procedures in the htp package. The function versions do not directly generate output in your web page. Instead, they pass their output as return values to the statements that invoked them. Use these functions when you need to nest calls. To print the output of htf functions, call them from within the htp.print procedure, which prints its parameter values to the generated web page.
owa	Contains subprograms required by the PL/SQL Gateway.
owa_content	Contains functions and procedures that let you query the content service repository and manipulate document properties.
owa_sec	Contains subprograms used by the PL/SQL Gateway for authenticating requests. **Note:** This package is included when you install the Toolkit with OAS. The PL/SQL Gateway does not use it.
owa_util	Contains utility subprograms. It is divided into the following areas: ■ Dynamic SQL utilities enable you to produce pages with dynamically generated SQL code. ■ HTML utilities enable you to retrieve the values of CGI environment variables and perform URL redirects. ■ Date utilities enable correct date-handling. Date values are simple strings in HTML, but are treated as a data type by the Oracle database.
owa_pattern	Contains subprograms to perform string matching and string manipulation with regular expression functionality.
owa_text	Contains subprograms used by owa_pattern for manipulating strings. They are externalized so you can use them directly.
owa_image	Contains subprograms that get the coordinates of where the user clicked on an image. Use this package when you have an image map whose destination links invoke the PL/SQL Gateway.

Package	Description
owa_cookie	Contains subprograms that enable you to send HTTP cookies to and get them from the client's browser. Cookies are opaque strings sent to the browser to maintain state between HTTP calls. State can be maintained throughout the client's session, or longer if an expiration date is included. Your system date is calculated with reference to the information specified in the owa_custom package.
owa_opt_lock	Contains subprograms that enable you to impose database optimistic locking strategies, so as to prevent lost updates. Lost updates can occur if a user selects and then updates a row whose values have been changed in the meantime by another user.
owa_custom	Contains the authorize function and the time zone constants used by cookies. **Note:** This package is included when you install the Toolkit with OAS. The PL/SQL Gateway does not use it.
owa_cache	Contains functions and procedures that let you use the PL/SQL Gateway cache feature to improve the performance of your PL/SQL web application.

5.2.1 htp and htf packages

The htp and htf packages provide subprograms that enable you to generate HTML tags from your stored procedure. For example, the following commands generate a simple HTML document:

```
create or replace procedure hello AS
BEGIN
    htp.htmlopen;           -- generates <HTML>
    htp.headopen;           -- generates <HEAD>
    htp.title('Hello');     -- generates <TITLE>Hello</TITLE>
    htp.headclose;          -- generates </HEAD>
    htp.bodyopen;           -- generates <BODY>
    htp.header(1, 'Hello'); -- generates <H1>Hello</H1>
    htp.bodyclose;          -- generates </BODY>
    htp.htmlclose;          -- generates </HTML>
END;
```

These packages also provide print procedures (such as htp.print), which writes its argument to the current document. You can use these print procedures to generate non-standard HTML, to display the return value of functions, or to pass hard-coded text that appears in the HTML document as-is. The generated text is passed to the PL/SQL Gateway, which then sends it to the user's browser.

5.2.2 owa_image package

The owa_image package contains subprograms that get the coordinates of where the user clicked on an image. You use this for image maps that invoke the PL/SQL Gateway. Your procedure would look something like:

```
create or replace procedure process_image
    (my_img in owa_image.point)
    as
    x integer := owa_image.get_x(my_img);
    y integer := owa_image.get_y(my_img);
begin
    /* process the coordinate */
end
```

5.2.3 owa_opt_lock

The owa_opt_lock package contains subprograms that enable you to impose database optimistic locking strategies, so as to prevent lost updates. Lost updates can occur if a user selects and then updates a row whose values have been changed in the meantime by another user.

The PL/SQL Gateway cannot use conventional database locking schemes because HTTP is a stateless protocol. The owa_opt_lock package gives you two ways of dealing with the lost update problem:

- The hidden fields method stores the previous values in hidden fields in the HTML page. When the user requests an update, the PL/SQL Gateway checks these values against the current state of the database. The update operation is performed only if the values match. To use this method, call the owa_opt_lock.store_values procedure.
- The checksum method stores a checksum rather than the values themselves. To use this method, call the owa_opt_lock.checksum function.

These methods are optimistic. That is, they do not prevent other users from performing updates, but they do reject the current update if an intervening update has occurred.

5.2.4 owa_custom

The owa_custom package contains the authorize function and the time zone constants used by cookies. Cookies use expiration dates defined in Greenwich Mean

Time (GMT). If you are not on GMT, you can specify your time zone using one of these two constants:

If your time zone is recognized, you can specify it directly using dbms_server_timezone. The value for this is a string abbreviation for your time zone. (See *Oracle Server SQL Reference* for a list of recognized time zones. For example, for Pacific Standard Time, use the following:

```
dbms_server_timezone constant varchar2(3) := 'PST'
```

If your time zone is not recognized, use dbms_server_gmtdiff to specify the offset of your time zone from GMT. Specify a positive number if your time zone is ahead of GMT, otherwise use a negative number.

```
dbms_server_gmtdiff constant number := NULL
```

After making the appropriate changes, reload the package.

5.2.5 owa_content

This owa_content package is included when you install the Toolkit with OAS. The PL/SQL Gateway does not use it.

The owa_content package contains functions and procedures that let you query the content service repository and manipulate document properties. Use this package to perform tasks such as:

- set a document description
- delete documents
- delete document attributes
- retrieve attribute information
- list document attributes
- retrieve content type of a document

When compiling PL/SQL procedures and packages that use the owa_content package, you may get the following error message:

```
PLS-00201
identifier 'WEBSYS.OWA_CONTENT' must be declared
```

To avoid this error, when creating a new DAD that uses a non local database, enter the SYS username and corresponding password when prompted for a DBA user. Entering the SYSTEM user does not allow the correct grant and rights to be

assigned to the database user. If you have entered SYSTEM as the DBA user then you must explicitly perform the grant privilege option as shown below:

```
SQL>grant all on WEBSYS.OWA_CONTENT to scott
```

If you are creating a DAD using an existing database user, you must perform the manual grant privilege shown above before using the OWA_CONTENT package.

Since the PL/SQL samples use the OWA_CONTENT package; these steps must be performed before installing the PL/SQL samples.

5.2.6 owa_cache

The owa_cache package contains functions and procedures that enable the PL/SQL Gateway cache feature to improve the performance of your PL/SQL web application. This section describes the specification of these functions and procedures. For more information about the PL/SQL Gateway cache feature, see "Caching" on page 1-19.

- owa_cache.disable

  Disables the cache for this particular request.

- owa_cache.set_expires(p_expires IN number, p_level IN varchar2)

  Sets up the cache headers for expires model cache type.

  Parameters:

  p_expires IN - the number of minutes this content is valid.

  p_level IN - the caching level for it.

  Exceptions:

  VALUE_ERROR is thrown if

  p_expires is negative or zero, or

  p_level is not 'USER' or 'SYSTEM', or

  p_expires is > 525600 (1 year).

- owa_cache.set_cache(p_etag IN varchar2, p_level IN varchar2)

  Sets up the cache headers for validation model cache type

  Parameters:

  p_etag IN - the tag associated with this content.

p_level IN - the caching level for it.

Exceptions:

VALUE_ERROR is thrown if

p_etag is greater than 55, or

p_level is not 'USER' or 'SYSTEM'.

- owa_cache.set_not_modified

  Sets up the headers for a not modified cache hit. Used in the Validation technique only (see "Validation technique" on page 1-20 for more information).

  Exceptions:

  VALUE_ERROR is thrown if **ETag** wasn't passed in.

- owa_cache.get_level

  Returns the caching level. Used in the Validation technique model only (see "Validation technique" on page 1-20 for more information).

  Returns:

  The caching level string ('USER' or 'SYSTEM') for cache hit; null otherwise.

- owa_cache.get_etag

  Returns the tag associated with the cached content. Used in the Validation technique only (see "Validation technique" on page 1-20 for more information).

  Returns:

  The tag for cache hit; null otherwise.

5.3 Conventions for parameter names in the toolkit

In the PL/SQL Web Toolkit, the first letter of the parameter name indicates the data type of the parameter:

Table 5–1

First character	Datatype	Example
c	VARCHAR2	cname IN VARCHAR2

Table 5–1

First character	Datatype	Example
n	INTEGER	nsize IN INTEGER
d	DATE	dbuf IN DATE

5.4 HTML tag attributes

Many HTML tags have a large number of optional attributes that, if passed as individual parameters to the hypertext procedures or functions, would make the calls cumbersome. In addition, some browsers support non-standard attributes. Therefore, each hypertext procedure or function that generates an HTML tag has as its last parameter `cattributes`, an optional parameter. This parameter enables you to pass the exact text of the desired HTML attributes to the PL/SQL procedure.

For example, the syntax for htp.em is:

```
htp.em(ctext, cattributes);
```

A call that uses HTML 3.0 attributes might look like the following:

```
htp.em('This is an example','ID="SGML_ID" LANG="en"');
```

which would generate the following:

```
<EM ID="SGML_ID" LANG="en">This is an example</EM>
```

5.5 PL/SQL Gateway and applets

When you reference an applet using the APPLET tag in an HTML file, the server looks for the applet class file in the directory containing the HTML file. If the applet class file is in another directory, use the CODEBASE attribute of the APPLET tag to specify that directory.

When you generate an HTML page from the PL/SQL Gateway and the page references an applet, specify the CODEBASE attribute because the PL/SQL Gateway does not have a concept of a current directory and does not know where to look for the applet class file.

The following example uses htp.appletopen to generate an APPLET tag. It uses the `cattributes` parameter to specify the CODEBASE value.

```
htp.appletopen('myapplet.class', 100, 200, 'CODEBASE="/applets"')
```

generates:

```
<APPLET CODE="myapplet.class" height=100 width=200 CODEBASE="/applets">
```

/applets is a virtual path that contains the myapplet.class file.

5.6 Cookies

Cookies can be used to maintain persistent state variables from the client browser:

```
http://home.netscape.com/newsref/std/cookie_spec.html
http://www.virtual.net/Projects/Cookies/
```

The owa_cookie package enables you to send and retrieve cookies in HTTP headers. It contains the following subprograms that you can use to set and get cookie values:

- owa_cookie.cookie data type contains cookie name-value pairs.
- owa_cookie.get function gets the value of the specified cookie.
- owa_cookie.get_all procedure gets all cookie name-value pairs.
- owa_cookie.remove procedure removes the specified cookie.

5.7 LONG Data Type

If you use values of the LONG data type in procedures/functions such as htp.print, htp.prn, htp.prints, htp.ps, or owa_util.cellsprint, be aware that only the first 32K of the LONG data is used. This reason for this limitation is that the LONG data is bound to a varchar2 data type in the procedure/function.

5.8 Extensions to the htp and htf Packages

The htp and htf packages allow you to use customized extensions. Therefore, as the HTML standard changes, you can add new functionality similar to the hypertext procedure and function packages to reflect those changes.

Here is an example of customized packages using non-standard <BLINK> and imaginary <SHOUT>tags:

```
create package nsf as
    function blink(cbuf in varchar2) return varchar2;
    function shout(cbuf in varchar2) return varchar2;
end;
```

```
create package body nsf as
    function blink(cbuf in varchar2) return varchar2 is
        begin return ('<BLINK>' || cbuf || '</BLINK>');
    end;
    function shout(cbuf in varchar2) return varchar2 is
        begin return ('<SHOUT>' || cbuf || '</SHOUT>');
    end;
end;

create package nsp as
    procedure blink(cbufin varchar2);
    procedure shout(cbufin varchar2);
end;

create package body nsp as
    procedure blink(cbufin varchar2) is
        begin htp.print(nsf.blink(cbuf));
    end;
    procedure shout(cbufin varchar2) is
        begin htp.print(nsf.shout(cbuf));
    end;
end;
```

Now you can begin to use these procedures and functions in your own procedure.

```
create procedure nonstandard as
begin
    nsp.blink('Gee this hurts my eyes!');
    htp.print('And I might ' || nsf.shout('get mad!'));
end;
```

5.9 String Matching and Manipulation

The owa_pattern package contains procedures and functions that you can use to perform string matching and string manipulation with regular expression functionality. The package provides the following subprograms:

- The owa_pattern.match function determines whether a regular expression exists in a string. It returns TRUE or FALSE.
- The owa_pattern.amatch function is a more sophisticated variation of the owa_pattern.match function. It lets you specify where in the string the match has to occur. This function returns the end of the location in the string where the

regular expression was found. If the regular expression is not found, it returns 0.

- The owa_pattern.change function and procedure lets you replace the portion of the string that matched the regular expression with a new string. If you call it as a function, it returns the number of times the regular expression was found and replaced.

These subprograms are overloaded. That is, there are several versions of each, distinguished by the parameters they use. Specifically, there are six versions of MATCH, and four each of AMATCH and CHANGE. The subprograms use the following parameters:

- line - This is the target to be examined for a match. It can be more than one line of text or a owa_text.multi_line data type.
- pat - This is the pattern that the subprograms attempt to locate in line. The pattern can contain regular expressions. In the owa_pattern.change function and procedure, this parameter is called from_str.
- flags - This specifies whether the search is case-sensitive or if substitutions are done globally.

5.10 owa_pattern.match

The regular expression in this function can be either a VARCHAR2 or a owa_pattern.pattern data type. You can create a owa_pattern.pattern data type from a string using the owa_pattern.getpat procedure.

You can create a multi_line data type from a long string using the owa_text.stream2multi procedure. If a multi_line is used, the rlist parameter specifies a list of chunks where matches were found.

If the line is a string and not a multi_line, you can add an optional output parameter called backrefs. This parameter is a row_list that holds each string in the target that was matched by a sequence of tokens in the regular expression. Here is an example of the owa_pattern.match function:

```
boolean foundMatch;
foundMatch := owa_pattern.match('KAZOO', 'zoo.*', 'i');
```

This is how the function works: KAZOO is the target where it is searching for the `zoo.*` regular expression. The period indicates any character other than newline,

and the asterisk matches 0 or more of the preceding characters. In this case, it matches any character other than the newline.

Therefore, this regular expression specifies that a matching target consists of zoo, followed by any set of characters neither ending in nor including a newline (which does not match the period). The i is a flag indicating that case is to be ignored in the search. In this case, the function returns TRUE, which indicates that a match had been found.

5.11 owa_pattern.change

owa_pattern.change can be a procedure or a function, depending on how it is invoked. As a function, it returns the number of changes made. If the flag 'g' is not used, this number can only be 0 or 1. The flag 'g' specifies that all matches are to be replaced by the regular expression. Otherwise, only the first match is replaced.

The replacement string can use the token ampersand (&), which indicates that the portion of the target that matched the regular expression is to be included in the expression that replaces it. For example:

```
owa_pattern.change('Cats in pajamas', 'C.+in', '& red ')
```

The regular expression matches the substring 'Cats in'. It then replaces this string with '& red'. The ampersand character, &, indicates 'Cats in', since that's what matched the regular expression. Thus, this procedure replaces the string 'Cats in pajamas' with 'Cats in red'. If you called this as a function instead of a procedure, the value it would return would not be 'Cats in red' but 1, indicating that a single substitution had been made.

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 3-2.
- 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. See "PL/SQL Web Toolkit Installation" on page 5-1 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

Setting up PL/SQL to run 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. See "Installing Required Packages" on page 3-2 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. See "Installing Required Packages" on page 3-2 for more information.

B

Troubleshooting

OAS Compatibility

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]
  [...]
  ```

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 1-8 for more information.

Logging

To turn on logging for mod_plsql, do the following:

1. Stop the Apache Listener.

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

2. Edit the file `<ORACLE_HOME>/Apache/modplsql/cfg/wdbsvr.app.`
3. Change the line containing `debugModules` to `debugModules=all`.
4. Restart your listener.

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
```

When you are finished debugging, revert the change in the configuration file (wdbsvr.app) by typing a semi-colon before the line to comment it out.Enabling logging degrades your web server's performance.

Controlling Database Processes for Each mod_plsql Request

The database connection pool in mod_plsql is not shared across Apache processes, which means that 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 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 MaxRequests = 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.
- Setup 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.

Debugging

The following series of steps can help you isolate your particular 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/owa_util.print_cgi_env
   ```

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.

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. As soon as you get this error, connect as SYS thru SQL*Plus, and issue "select username from v$session". If the count of the number of rows is almost the same as what you have in init$SID.ora, then you are having this problem.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N