Oracle® Application Server 10*g*

mod_plsql User's Guide

10*g* (9.0.4)

Part No. B10357-01

September 2003

Oracle Application Server 10*g* mod_plsql User's Guide, 10*g* (9.0.4)

Part No. B10357-01

Primary Author: Peter Lubbers

Contributors: Pravin Prabhakar, Pushkar Kapasi, Eric Lee, Cheryl Smith, Sanjay Khanna, Ron Decker, and Kannan Muthukkaruppan.

Contents

Send Us Your Comments

Oracle Application Server 10*g* mod_plsql User's Guide, 10*g* (9.0.4)

Part No. B10357-01

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

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

If you find any errors in this document, or have suggestions for improvement, please send your comments to appserverdocs_us@oracle.com. Please indicate the title and part number of the documentation and the chapter, section, and page number (if available). If you would like a reply, please give your name, address, telephone number, and email address.

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

Preface

This manual describes how to install, configure, and maintain mod_plsql for Oracle Application Server 10*g* (9.0.4). It contains the following chapters:

- Chapter 1 - Provides an overview of the mod_plsql and its features.
- Chapter 2 - Explains how to use mod_plsql.
- Chapter 3 - Describes how to secure the database and PL/SQL using mod_plsql.
- Appendix A - Frequently Asked Questions

New Features

- Enhanced cache clean up algorithm allowing users to configure the time to clean up cache.
- Enhanced OWA packages to allow better byte-packing of response data from multi-byte databases resulting in lesser round-trips to the database
- Reduced PL/SQL function call overheads in heavily called OWA package APIs like **htp.p**.
- Enhanced performance logging in mod_plsql. See "What kind of logging facilities are available in mod_plsql?" in Appendix A, "Frequently Asked Questions" for more information.
- Enhanced security by obscuring schema passwords in the DAD configuration.
- Integration with Enterprise Manager for log analysis and log correlation.

See Also: *Oracle Application Server 10g Administrator's Guide*

- Dropped requirement to have `/pls` in the virtual path for accessing mod_plsql. It can still be used, but is no longer required. mod_plsql has been integrated in the Oracle HTTP Server configuration since release 9.0.x, and therefore, it no longer has any dependency on having "`/pls/DAD`" as the entry point for mod_plsql. Note that OracleAS Portal DADs still need to be of the format `/pls/dad`.

Related Documentation

You may also find the following manuals in the Oracle Application Server documentation set useful:

Title	Containing Information about...
Oracle HTTP Server Administrator's Guide	Oracle HTTP Server Modules - Administration using the command line tools and manually editing configuration files (DAD parameters, http.conf etc.)
Oracle Application Server 10g Performance Guide	Performance and tuning material and caching
Oracle Application Server 10g Administrator's Guide	Administering Oracle Application Server through the Oracle Enterprise Manager Console (DAD Configuration)
Oracle Application Server 10g Upgrading to 10g (9.0.4)	Parameters that have changed and tools used to migrate DADs
Oracle Application Server 10g PL/SQL Web Toolkit Reference	OWA package information
Oracle Application Server 10g Installation Guide	Installation for the Oracle Application Server
Oracle Application Server 10g Migrating from Oracle Application Server	Migrating from previous versions (Oracle Application Server)
Oracle Application Server 10g Concepts	Overview of the Oracle Application Server

Documentation Accessibility

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

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

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Conventions

The following conventions are used in this manual:

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

Oracle Services and Support

Information about Oracle products and global services is available from:

`http://www.oracle.com`

The sections below provide URLs for selected services.

Oracle Technology Network

Register with the Oracle Technology Network (OTN) at:

`http://otn.oracle.com`

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

Oracle Support Services

Technical Support contact information worldwide is listed at:

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

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

Product and Documentation

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

`http://store.oracle.com`

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

Product documentation can be found at:

`http://docs.oracle.com`

Customer Service

Global Customer Service contacts are listed at:

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

Education and Training

Training information and worldwide schedules are available from:

`http://education.oracle.com`

1 Understanding mod_plsql

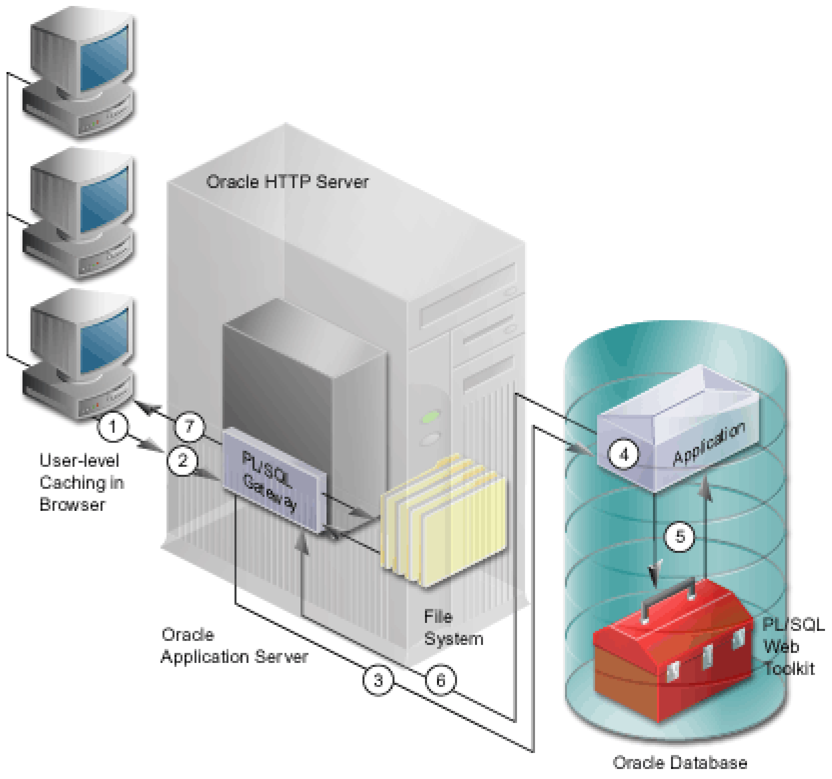
Oracle Application Server consolidates Oracle's middle-tier products into a single solution for the deployment of Web applications. mod_plsql provides support for building PL/SQL-based applications on the Web. PL/SQL stored procedures retrieve data from a database and generate HTTP responses containing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

This chapter discusses the following topics:

1.1 Processing Client Requests

mod_plsql is an Oracle HTTP Server plug-in that communicates with the database. It maps browser requests into database stored procedure calls over a SQL*Net connection. It is often indicated by a `/pls` virtual path.

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



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

7. The Oracle HTTP Server sends the response to the client browser.

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

1.2 Database Access Descriptors (DADs)

Each mod_plsql 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.

> **See also:** *Oracle HTTP Server Administrator's Guide* for descriptions of the DAD parameters and an overview of the mod_plsql configuration files.

1.3 Invoking mod_plsql

To invoke mod_plsql in a Web browser, input the URL in the following format:

```
protocol://hostname[:port]/DAD_location/[[!][schema.][package.]proc_name[?query_string]]
```

Table 1–1 Invoking mod_plsql 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.

Table 1–1 Invoking mod_plsql Parameters

Parameter	Description
DAD location	A virtual path to handle PL/SQL requests that you have configured in the Web server. The DAD location can contain only ASCII characters.
! character (optional)	Indicates to use the flexible parameter passing scheme. See Section 1.6.2, "Flexible Parameter Passing" for more information.
schema (optional)	The database schema name. If omitted, name resolution for *package.proc_name* occurs based on the database user that the URL request is processed as.
package (optional)	The package that contains the PL/SQL stored procedure. If omitted, the procedure is stand-alone.
proc_name	The PL/SQL stored procedure to run. This must be a procedure and not a function. It can accept only IN arguments.
?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 mod_plsql. See "POST, GET and HEAD Methods" on page 1-5 for more information.

Example 1–1 A Web server is configured with `/pls/mydad` as a DAD location 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 mod_plsql. This is because the `/pls/mydad` indicates that the Web server is configured to invoke mod_plsql. It then uses the DAD associated with `/pls/mydad` and runs the `myproc` procedure stored in `mypackage`.

Example 1–2 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 DAD Configuration pages) displays.

> **Note:** When a DAD parameter in the `dads.conf` file is misconfigured, a log message is written out to Oracle HTTP Server's `error_log` file, and that DAD will be disabled. A HTTP 500 status error is displayed to a user who is accessing that DAD from a browser.

POST, GET and HEAD Methods

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

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

- When using the POST method, parameters are passed in the request body. Generally, if you are passing large amounts of parameter data to the server, use the POST method.
- When using the GET method, parameters are passed using a query string. The limitation of this method is that the length of the value in a name-value pair cannot exceed the maximum length for the value of an environment variable, as imposed by the underlying operating system. In addition, operating systems have a limit on how many environment variables you can define.
- When using the HEAD method, it has the same functionality as the GET method. The only difference is that only the HTTP status line and the HTTP headers are passed back. No content data is streamed back to the browser. This is useful for monitoring tools in which you are only interested if the request is processed correctly.
- Mixed Mode - In mod_plsql you can pass some of the parameters in a query string and the remaining ones as POST data. For example, if you have a procedure foo (a varchar2, b number), and want to pass values "v" and "1" to 'a' and 'b' respectively, you could do so in three ways to create URLs:
  - All values are specified as part of the query string.

    ```
    http://host:port/pls/DAD/foo?a=v&b=1
    ```
  - All values are specified as part of the POST data.

```
http://host:port/pls/DAD/foo, POST data="a=v&b=1"
```

- Some of the parameters are specified in the URL and the rest in the POST data.

```
http://host:port/pls/DAD/foo?a=v, POST data="b=1"
```

1.4 Transaction Mode

After processing a URL request for a procedure invocation, mod_plsql performs a rollback if there were any errors. Otherwise, it 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.

1.5 Supported Data Types

Because HTTP supports character streams only, mod_plsql supports the following subset of PL/SQL data types:

- NUMBER
- VARCHAR2
- TABLE OF NUMBER
- TABLE OF VARCHAR2

Records are not supported.

1.6 Parameter Passing

mod_plsql 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 Section 1.6.1, "Parameter Passing by Name (Overloaded Parameters)" for more information.

- Flexible parameter passing

  Procedures are prefixed by a ! character. See Section 1.6.2, "Flexible Parameter Passing" for more information.

- Large (up to 32K) parameters passing

  See Section 1.6.3, "Large Parameter Passing" for more information.

Note: mod_plsql handles multi-value variables by storing the values in a PL/SQL table. This enables you to be flexible about how many values the user can pick, and it makes it easy for you to process the user's selections as a unit. Each value is stored in a row in the PL/SQL table, starting at index 1. The first value (in the order that it appears in the query string) of a variable that has multiple values is placed at index 1, the second value of the same variable is placed at index 2, and so on. If the order of the values in the PL/SQL table is significant in your procedure, you need to determine the order in which the variables appear in the query string, or modify your PL/SQL application to do the ordering internally.

If you do not have variables with multiple values, the order in which the variables appear does not matter, because their values are passed to the procedure's parameters by name, and not by position.

The PL/SQL tables used as parameters in the mod_plsql environment must have a base type of `VARCHAR2`. Oracle can convert `VARCHAR2` to other data types such as `NUMBER`, `DATE`, or `LONG`. The maximum length of a `VARCHAR2` variable is 32K.

If you cannot guarantee that at least one value will be submitted to the PL/SQL table (for example, the user can select no options), use a hidden form element to provide the first value. Not providing a value for the PL/SQL table produces an error, and you cannot provide a default value for a PL/SQL table.

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.

You must give parameters different names for overloaded subprograms that have the same number of parameters. Because HTML data is not associated with datatypes, mod_plsql 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 mod_plsql.

```
-- legal PL/SQL, but not for mod_plsql
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 mod_plsql
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, mod_plsql 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 mod_plsql gets a request that has only one value for the val parameter, it invokes the procedure with the scalar data type.

Example 1–3 Send the following URL to execute the scalar version of the procedure:

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

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

Example 1–4 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

You can have HTML forms from which users can select any number of elements. If these elements have different names, you would have to create overloaded procedures to handle each possible combination. Alternatively, you could insert hidden form elements to ensure that the names in the query string are consistent each time, regardless of what elements the user chooses. mod_plsql makes this operation easier by supporting 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 mod_plsql. The four parameter interface is supported for compatibility.

1.6.2.1 Two Parameter Interface

```
procedure [proc_name]
     (name_array IN [array_type],
     value_array IN  [array_type]);
```

Table 1–2 Two Parameter Interface Parameters

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

Table 1–2 Two Parameter Interface Parameters

Parameter	**Description**
array_type (required)	Any PL/SQL index-by table of varchar2 type (e.g., owa.vc_arr).

Example 1–5 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 mod_plsql to use flexible parameter passing. It invokes procedure *scott.myproc* and passes it the following two arguments:

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

Note: When using this style of Flexible Parameter Passing in Oracle Application Server, the procedure must be defined with the parameters *name_array* and *value_array*. The datatypes of these arguments should match the datatypes shown in the example.

1.6.2.2 Four Parameter Interface

The four parameter interface is supported for compatibility.

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

Table 1–3 Four Parameter Interface Parameters

Parameter	**Description**
proc_name (required)	The name of the PL/SQL procedure that you are invoking.
num_entries	The number of name_value pairs in the query string
name_array	The names from the query string (indexed from 1) in the order submitted.

Table 1–3 Four Parameter Interface Parameters

Parameter	Description
value_array	The values from the query string (indexed from 1) in the order submitted.
reserved	Not used. It is reserved for future use.
array_type (required)	Any PL/SQL index-by table of varchar2 type (e.g., owa.vc_arr).

Example 1–6 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 mod_plsql to use flexible parameter passing. It invokes procedure `scott.my_pkg.myproc` and passes it the following arguments:

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

Note: When using this style of Flexible Parameter Passing in Oracle Application Server, the procedure must be defined with the parameters *num_entries*, *name_array*, *value_array*, and *reserved*. The datatypes of these arguments should match the datatypes shown in the example.

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 Section 1.6.2, "Flexible Parameter Passing"), 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

mod_plsql 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/pls/mydad/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 (Binary Large Object) types of columns in the document table.

This section discusses the following:

- Document Table Definition
- Old Style Document Table Definition
- Configuration Parameters for Document Upload/Downloading
- File Upload
- Specifying Attributes (Mime Types) of Uploaded Files
- Uploading Multiple Files
- File Download
- Direct BLOB Download

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:

```
CREATE TABLE MYDOCTABLE (
  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, mod_plsql 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, mod_plsql 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.

mod_plsql 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, mod_plsql 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 Configuration Parameters for Document Upload/Downloading

The following configuration parameters in the DAD affect a document upload/download operation:

- "PlsqlDocumentTablename"
- "PlsqlDocumentPath (Document Access Path)"

- "PlsqlDocumentProcedure (Document Access Procedure)"
- "PlsqlUploadAsLongRaw"

Example 1–7   Parameters for Document Upload/Download

If the configuration for these parameters in a DAD is as follows:

```
PlsqlDocumentTablename   scott.my_document_table
PlsqlUploadAsLongRaw   html
PlsqlDocumentPath   docs
PlsqlDocumentProcedure   scott.my_doc_download_procedure
```

then:

- mod_plsql will retrieve data from, or store to a database table called **my_document_table** in the **scott** schema.
- All file extensions except `.html` will be uploaded to the document table as BLOBs. All files with `.html` extension will be uploaded as *Long Raw*.
- All URLs which have the keyword **docs** immediately following the DAD location will result in invocation of the procedure **scott.my_doc_download_procedure**.

  Typically, this procedure will call **wpg_docload.download_file** to initiate a file download for a file whose name is based on the URL specification.

A simple example with the above configuration is:

```
http://www.acme.com/pls/dad/docs/index.html
```

This results in downloading of the file `index.html` from the *Long Raw* column of the database table **scott.my_document_table**. Note that the application procedure has full control on the file download to initiate, and has the flexibility to define a more complex *PlsqlDocumentProcedure* that implements file-level access controls and versioning.

Note: The application defined procedure **scott.my_doc_download_procedure** has to be defined without arguments, and should rely on the CGI environment variables to process the request.

1.7.3.1 PlsqlDocumentTablename

The `PlsqlDocumentTablename` parameter specifies the table for storing documents when file uploads are performed via this DAD.

Syntax:

```
PlsqlDocumentTablename  [document_table_name]
```

```
PlsqlDocumentTablename  my_documents
```
or,

```
PlsqlDocumentTablename  scott.my_document_table
```

1.7.3.2 PlsqlDocumentPath (Document Access Path)

The `PlsqlDocumentPath` parameter specifies the path element to access a document. The `PlsqlDocumentPath` 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://www.acme.com/pls/mydad/docs/myfile.htm
```

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

Syntax:

```
PlsqlDocumentPath  [document_access_path_name]
```

1.7.3.3 PlsqlDocumentProcedure (Document Access Procedure)

The `PlsqlDocumentProcedure` 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 a file. It knows the filename based on the URL specification. For example, an application can use this to implement file-level access controls and versioning. An example of this is in Section 1.7.7, "File Download".

Syntax:

```
PlsqlDocumentProcedure  [document_access_procedure_name]
```

Example 1–8

```
PlsqlDocumentProcedure  my_access_procedure
```
or,

```
PlsqlDocumentProcedure  scott.my_pkg.my_access_procedure
```

1.7.3.4 PlsqlUploadAsLongRaw

The DAD parameter, `PlsqlUploadAsLongRaw`, configures file uploads based on their file extensions. The value of an `PlsqlUploadAsLongRaw` DAD parameter is a one entry per line list of file extensions. Files with these extensions are uploaded by mod_plsql 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.) or an asterisk (*) matches any file whose extension has not been listed in the `PlsqlUploadAsLongRaw` setting.

Syntax:

```
PlsqlUploadAsLongRaw  [file_extension]
PlsqlUploadAsLongRaw *
```

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

Example 1–9

```
PlsqlUploadAsLongRaw  html
PlsqlUploadAsLongRaw  txt
PlsqlUploadAsLongRaw  *
```

1.7.4 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 mod_plsql 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. mod_plsql stores the file contents in the database in the document storage table. The table name is derived from the PlsqlDocumentTablename DAD setting.

3. The action procedure specified in the *action* attribute of the FORM is run (similar to invoking a mod_plsql procedure without file upload).

> **Note:** The parsing of HTML documents is deprecated in mod_plsql. In Oracle Application Server version 1.0.2.2 and below, mod_plsql used to parse the content of an HTML file when it was uploaded, and identified other files that the HTML document was referring to. This information was then stored into a table. The table name was constructed by appending the name of the document table with "part". This functionality was found to be not of use to customers and has been deprecated in the versions of mod_plsql released since Oracle Application Server v1.0.2.

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 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:

a. The browser uploads the file listed in the `INPUT type="file"` element.

b. The `write_info` procedure then runs.

c. The procedure writes information from the form fields to a table in the database and returns a page to the user.

Note: 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, as shown below.

```
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 mod_plsql is `F9080/private/minutes.txt.` The application can rename this in the called stored procedure. For example, the application can rename it to `scott/minutes.txt`.

1.7.5 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 Section 1.7.4, "File Upload" 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 mod_plsql when uploading the file.

1.7.6 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).
```

1.7.7 File Download

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

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

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

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

```
http://www.acme.com:9000/pls/mydad/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 mod_plsql, a download of the file *`filename`* is initiated. However, when a file download begins, no other HTML (produced via HTP interfaces) generated by the procedure, is passed back to the browser.

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

1.7.8 Direct BLOB Download

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

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

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

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

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

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

   b. Setup headers which describes the content

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

```
owa_util.http_header_close;
```

c. Initiate Direct BLOB download

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

The structure of the mytable table:

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

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

1.8 Path Aliasing (Direct Access URLs)

Path Aliasing enables applications using mod_plsql to provide direct reference to its objects using simple URLs. The Path Aliasing functionality is a generalization of how the document download functionality is provided. The following configuration parameters in the DAD are used for Path Aliasing:

- PlsqlPathAlias
- PlsqlPathAliasProcedure

For Example, if the configuration for these parameters in a DAD is as follows:

```
PlsqlPathAlias   myalias
PlsqlPathAliasProcedure   scott.my_path_alias_procedure
```

then, all URLs which have the keyword **myalias** immediately following the DAD location will invoke the procedure **scott.my_path_alias_procedure**. Based on the URL specification, this procedure can initiate an appropriate response.

Note: The application defined procedure **scott.my_path_alias_procedure** has to be defined to take one argument of type *varchar2* called **p_path**. This argument will receive everything following the keyword used in *PlsqlPathAlias*.

For example, in the above configuration, the URL:

```
http://www.acme.com/pls/dad/myalias/MyFolder/MyIte
m
```

will result in the procedure **scott.my_path_alias_procedure** receiving the argument **MyFolder/MyItem**.

1.9 Common Gateway Interface (CGI) Environment Variables

The OWA_UTIL package provides an API to get the values of CGI environment variables. The variables provide context to the procedure being executed through mod_plsql. Although mod_plsql is not operated through CGI, the PL/SQL application invoked from mod_plsql can access these CGI environment variables.

Given below is the list of CGI environment variables:

- HTTP_AUTHORIZATION
- DAD_NAME
- DOC_ACCESS_PATH
- HTTP_ACCEPT
- HTTP_ACCEPT_CHARSET
- HTTP_ACCEPT_LANGUAGE
- HTTP_COOKIE
- HTTP_HOST
- HTTP_PRAGMA
- HTTP_REFERER
- HTTP_USER_AGENT
- PATH_ALIAS
- PATH_INFO
- HTTP_ORACLE_ECID

- DOCUMENT_TABLE
- REMOTE_ADDR
- REMOTE_HOST
- REMOTE_USER
- REQUEST_CHARSET (refer to Section 1.9.2.1, "REQUEST_CHARSET CGI Environment Variable")
- REQUEST_IANA_CHARSET (refer to Section 1.9.2.2, "REQUEST_IANA_CHARSET CGI Environment Variable")
- REQUEST_METHOD
- REQUEST_PROTOCOL
- SCRIPT_NAME
- SCRIPT_PREFIX
- SERVER_NAME
- SERVER_PORT
- SERVER_PROTOCOL

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

Syntax:

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

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

1.9.1 Adding and Overiding CGI Environment Variables

The `PlsqlCGIEnvironmentList` DAD parameter is a one-entry per line 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 Section 1.9, "Common Gateway Interface (CGI) Environment Variables"), 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.

Note: Refer to the *Oracle HTTP Server Administrator's Guide* for information about the mod_plsql Configuration Files.

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

```
PlsqlCGIEnvironmentList DOCUMENT_ROOT
```

New environment variables passed in through this configuration parameter are available to the PL/SQL application via the owa_util.get_cgi_env interface.

Example 1–10

```
PlsqlCGIEnvironmentList SERVER_NAME=myhost.mycompany.com
PlsqlCGIEnvironmentList 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 1–11

```
PlsqlCGIEnvironmentList MYENV_VAR=testing
PlsqlCGIEnvironmentList SERVER_NAME=
PlsqlCGIEnvironmentList REMOTE_USER=user2
```

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

1.9.2 PlsqlNLSLanguage

For mod_plsql, the National Language Support variable (*PlsqlNLSLanguage*) can be set either as an environment variable or at the DAD level. The following restrictions apply:

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

If *PlsqlNLSLanguage* is not configured at the DAD level, the NLS setting is picked up from the environment - either from the setting of NLS_LANG in `ORACLE_HOME/opmn/conf/opmn.xml`. If that does not exist, the default rules apply for NLS_LANG settings for Oracle.

1.9.2.1 REQUEST_CHARSET CGI Environment Variable

Every request to mod_plsql 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, derived from the PlsqlNLSLanguage environment variable. However, if the DAD level PlsqlNLSLanguage parameter is set, that derives the character set information instead.

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

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

1.9.2.2 REQUEST_IANA_CHARSET CGI Environment Variable

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

1.10 Restrictions in mod_plsql

The following restrictions exist in mod_plsql:

- The maximum length of the HTTP cookie header is 32000 bytes. Values higher than this generate an error. This limit is due to the PL/SQL varchar2 limit.
- The maximum length of any single cookie within the HTTP cookie is 3990. Values higher than this generate an error. This limit is due to the OCI array bind limit of strings in arrays.
- There is a hard maximum cookie limit in mod_plsql that limits the number of cookies being set at any given time. That limit is set to 20. Anything over 20 will be dropped.
- The PL/SQL Gateway does not support calling procedures with OUT paramaters to be called from a Web interface. Doing this may result in ORA-6502 errors. The recommended approach is not to call any procedure that has OUT variables in it. However, the current architecture will let you modify a value as long as the modified value does not exceed the length that was passed

in. Existing applications that encounter this problem need to be modified in one of the following ways:

  - Implement wrappers for procedures with OUT parameters so that such procedures are not invoked directly via a browser URL.
  - Create a local variable that gets assigned the value of the parameter being passed in, and is then used for all internal changes.

- The total number of name value pairs that can be passed to a PL/SQL procedure is 2000.
- mod_plsql limits the total number of parameters that can be passed to a single procedure to 2000.
- mod_plsql limits the size of a single parameter that can be passed to a procedure to 32000 bytes.
- It is not possible to use identical DAD locations in different virtual hosts.

2 Using mod_plsql

This chapter describes how you can set up and use mod_plsql. It contains the following sections:

2.1 Before You Begin

Before you run mod_plsql, you must satisfy the following requirements:

- You must have a SYS user password on the database where you plan to load PL/SQL Web Toolkit packages required by mod_plsql.
- The database to which you plan to connect mod_plsql must be up and running.
- Oracle Application Server version 10*g* (9.0.4) ships with OWA package version 9.0.4.0.1. It is recommended that the OWA packages installed in the database are at least version 9.0.4.0.0.

2.2 Installing Required Packages

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

Note: Even if a full database export is made with the Export utility you still must reinstall mod_plsql in the new target instance via running the `OWALOAD.SQL` script as SYS. Objects in SYS are not imported with the Import/Export mechanism, and the PL/SQL toolkit has to be installed in SYS.

---

1. Navigate to the directory where the owaload.sql file is located. This directory is *`ORACLE_HOME`*`/Apache/modpsql/owa.`
2. Using SQL*Plus, log into the Oracle database as the SYS user.
3. Oracle Application Server 10*g* (9.0.4) ships with OWA package version 9.0.4.0.1. Check the version of the OWA packages currently installed by running the following query:

   ```
   select owa_util.get_version from dual;
   ```

   - If the query succeeds, but shows a version less than 9.0.4.0.1, it is recommended that you install the newer OWA packages.
   - If the query fails, you either do not have the OWA packages installed, or are running a very old version of OWA packages, and it is recommended that you install, or upgrade to the new OWA packages.

Note: To detect older OWA packages, see "How do I detect and clean up duplicate OWA packages installed in the database?" in Appendix A, "Frequently Asked Questions".

---

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

   ```
   @owaload.sql log_file
   ```

Table 2–1 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. Make sure that you have write permissions to create the log file

5. Scan the log file for any errors.

   Note: The `owaload` script checks the existing version of the OWA packages in the database and installs a new version only if:

   - No OWA package exists or,
   - Older OWA packages were detected. If your database already has the latest OWA packages or has a newer version installed, the `owaload` script does nothing and reports this in the log file.

6. Do a manual recompile.

   Note: Installing the OWA packages invalidates all dependent objects. These packages automatically recompile on first access, but a manual recompile is recommended after the reinstallation.

   After the install, check the version of the OWA packages by running "`Select owa_util.get_version from dual;`". Confirm that the version shown is 9.0.4.0.1 or above.

7. Note that public access is now granted to:

   - OWA_CUSTOM
   - OWA
   - HTF
   - HTP
   - OWA_COOKIE
   - OWA_IMAGE
   - OWA_OPT_LOCK
   - OWA_PATTERN
   - OWA_SEC
   - OWA_TEXT
   - OWA_UTIL
   - OWA_CACHE

- WPG_DOCLOAD

8. Note also that the following public synonyms are created:
    - OWA_CUSTOM for OWA_CUSTOM
    - OWA_GLOBAL for OWA_CUSTOM
    - OWA for OWA
    - HTF for HTF
    - HTP for HTP
    - OWA_COOKIE for OWA_COOKIE
    - OWA_IMAGE for OWA_IMAGE
    - OWA_OPT_LOCK for OWA_OPT_LOCK
    - OWA_PATTERN for OWA_PATTERN
    - OWA_SEC for OWA_SEC
    - OWA_TEXT for OWA_TEXT
    - OWA_UTIL for OWA_UTIL
    - OWA_INIT for OWA_CUSTOM
    - OWA_CACHE for OWA_CACHE
    - WPG_DOCLOAD for WPG_DOCLOAD

2.2.1 Upgrading from Oracle Application Server or WebDB Listener

If you were previously running Oracle 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.
3. Install the new PL/SQL Web Toolkit as per Section 2.2, "Installing Required Packages".

2.3 Accessing the mod_plsql Configuration Page

All monitoring and configuration takes place through the Oracle Enterprise Manager (OEM) tool. The mod_plsql monitoring and configuration is accessible through links within the HTTP Server components and the Portal.

Note: Refer also to the Online Help available through the Oracle Enterprise Manager tool for mod_plsql and DAD Configuration.

To get to the mod_plsql configuration pages navigate to the applicable Application Server through OEM. You can access the pages either through a portal instance or through the link in the HTTP server instance.

2.3.1 Access the DAD Configuration Pages Through Oracle Enterprise Manager

1. Enter the following URL in a Web browser:

   ```
   http://<hostname>:<port_number>
   ```

   Note: 1810 is the default port.

   ---

2. Enter the Oracle Application Server administrator username and password. The default username for administrator user is ias_admin. The default password is defined during the installation of Oracle Application Server.
3. Click **OK**.
4. Select Oracle Application Server instance with the mod_plsql that needs configuring.
5. Select the HTTP Server link or the Portal instance.
6. Select mod_plsql component or link.
7. Scroll down to **DAD Status** section.
8. Click **Create** to set up a new DAD, or click the name of the DAD you are interested in.

2.3.2 Access the DAD Configuration Pages Through Portal

1. Log on to Portal.

2. Click the **Builder** icon.
3. Access the **Administer** tab.
4. Click **Portal Service Monitoring** link in the Services portlet.
5. Click **mod_plsql Services** in the Portal Components section.
6. Scroll down to **DAD Status** section.
7. Click **Create** to set up a new DAD, or click the name of the DAD you are interested in.

2.4 Configuring mod_plsql for Use with OracleAS Portal 3.0.9

To run an Oracle Application Server version 9.0.x middle-tier against a OracleAS Portal 3.0.9 repository, you need to perform the following steps:

1. Use Oracle Enterprise Manager to create two Portal-style DADs: one for the 3.0.9 OracleAS Portal repository and another for the 3.0.9 Oracle Application Server Single Sign-On repository.

   Note: In Oracle Application Server version 9.0.x, the DAD name is case-sensitive. You must ensure that the DAD is created with the same case as it exists for OracleAS Portal version 3.0.9. Mostly, the DAD name is all in lowercase. To see a version 3.0.9 DAD name, view the file `$OLD_HOME/Apache/modplsql/cfg/wdbsvr.app`.

2. For each DAD, edit the DAD configuration and set *PlsqlCompatibilityMode* to `1`. To edit the DAD configuration:

   a. Edit the file *`ORACLE_HOME`*`/Apache/modplsql/conf/dads.conf`.

   b. Locate the DADs used to connect to the version 3.0.9 OracleAS Portal and Oracle Application Server Single Sign-On repositories.

   c. Add the following line in the DADs:

   ```
   PlsqlCompatibilityMode  1
   ```

   d. Run the following command:

   ```
   ORACLE_HOME/dcm/bin/dcmctl updateConfig -ct ohs
   ```

   e. Run the following command to restart Oracle HTTP Server:

```
ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
```

Note:

- If *PlsqlCompatibilityMode* is not set for a DAD which is used to access an OracleAS Portal version 3.0.9 repository, you will be unable to download documents that have spaces in the document name, and reside in the Portal document table. Access to such documents will result in the error: **HTTP 404 - File not found**.
- If *PlsqlCompatibilityMode* is set for a DAD which is used to access an OracleAS Portal version 9.0.x repository, you will be unable to download documents that have spaces or plus (+) signs in the document name, and reside in the Portal document table. Access to such documents will result in the error: **HTTP 404 - File not found**.

Warning: Once the back-end OracleAS Portal repository is upgraded to version 9.0.x, you should remove the *PlsqlCompatibilityMode* flag from the DAD. See the section "Removing the PlsqlCompatibilityMode Flag from a DAD" for detailed instructions.

Removing the PlsqlCompatibilityMode Flag from a DAD

To remove the *PlsqlCompatibilityMode* flag from a DAD, after the back-end OracleAS Portal repository is upgraded to version 9.0.x, perform the following steps:

1. Edit the file `ORACLE_HOME/Apache/modplsql/conf/dads.conf`.
2. Locate the DADs used to connect to the version 9.0.x OracleAS Portal and Oracle Application Server Single Sign-On repositories.
3. Remove the following line in the DAD:

   ```
   PlsqlCompatibilityMode  1
   ```

4. Run the following command:

   ```
   ORACLE_HOME/dcm/bin/dcmctl updateConfig -ct ohs
   ```

5. Run the following command to restart Oracle HTTP Server:

   ```
   ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
   ```

3 Securing Application Database Access Through mod_plsql

This chapter describes how to set up the database and PL/SQL to avoid known security problems. It covers the following topics:

- Authenticating Users Through mod_plsql
- Deauthenticating Users
- Protecting the PL/SQL Procedures Granted to PUBLIC

> **See Also:** For more information about mod_plsql, refer to the *Oracle Application Server 10g PL/SQL Web Toolkit Reference* in the Oracle Application Server Documentation Library.

3.1 Authenticating Users Through mod_plsql

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

You can enable different authentication modes, as described in Table 3–1.

Table 3–1 Authentication Modes Used with mod_plsql

Authentication Mode	Approach
Basic	Authentication is performed using basic HTTP authentication. Most applications use basic authentication.

Table 3–1 Authentication Modes Used with mod_plsql

Authentication Mode	Approach
Global OWA	Authentication is performed using the owa_custom.authorize procedure in the schema containing the PL/SQL Web Toolkit packages.
Custom OWA	Authentication is performed using packages and procedures in the user's schema (owa_customize.authorize), 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 (packageName.authorize).
Single Sign-on	Authentication is performed using Oracle Application Server Single Sign-On. Use this mode only if your application works with Oracle Application Server Single Sign-On.

3.1.1 Basic (Database Controlled Authentication)

The module, mod_plsql, 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 by means of the browser's Basic HTTP Authentication dialog box. The user must provide a user name and password in the dialog box.

3.1.2 Oracle Application Server Basic Authentication Mode

Oracle Application Server has a different mechanism for the basic authentication mode. The user name and password must be stored in the DAD. Oracle Application Server 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.

Oracle Application Server Basic Authentication Mode

mod_plsql supports Oracle Application Server basic authentication. Oracle HTTP Server authenticates users' credentials against a password file on the file system. This functionality is provided by a module called `mod_auth`.

3.1.3 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. Custom authentication uses a static user name and password that is stored in the DAD. It cannot be combined with dynamic user name and password authentication.

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

- Global OWA enables you to invoke the same authentication function for all users and procedures.
- Custom OWA enables you to invoke a different authentication function for each user and for all procedures.

Per Package authentication enables you to invoke the authentication function for all users, but only for procedures in a specific package or for anonymous procedures.

Table 3–2 summarizes the parameter values.

Table 3–2 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 packages	`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.

3.2 Deauthenticating Users

For DADs using dynamic authentication (no username/password in the DAD), mod_plsql allows users to log off (clear HTTP authentication information) programmatically through a PL/SQL procedure without having to exit all browser instances. This feature is supported on Netscape 3.0 or higher and on Microsoft Internet Explorer. On other browsers, the user may have to exit the browser to deauthenticate.

Deauthentication can be done programatically by creating your own logout procedure which simulates a logout and redirects the user to a sign-off page.

Create or replace procedure *MyLogOffProc* as follows:

```
BEGIN
   -- Open the HTTP header
   owa_util.mime_header('text/html', FALSE, NULL);

   -- Send a cookie to logout
   owa_cookie.send('WDB_GATEWAY_LOGOUT', 'YES', path=>'/');

   -- Close the HTTP header
   owa_util.http_header_close;

   -- Generate the page
   htp.p('You have been logged off from the WEBSITE');
   htp.anchor( 'http://www.abc.com', 'click here');
   htp.p('<BR>bye');
END;
```

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

```
http://www.abc.com:2000/pls/myDAD/logmeoff
```

3.3 Protecting the PL/SQL Procedures Granted to PUBLIC

Every database package granted to public can be directly executed using the following URL:

```
http://hostname:port/pls/dad/schema.package.procedure
```

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 vulnerability when they are executed through a Web browser. Such packages are intended only for the PL/SQL application developer.

3.3.1 Using the PlsqlExclusionList Directive in mod_plsql

mod_plsql provides a DAD parameter directive called *PlsqlExclusionList* to protect the execution of these PL/SQL packages and other packages that are specific to applications. The *PlsqlExclusionList* directive specifies a pattern for procedure, package, and schema names that are forbidden to be directly executed from a browser. This is a multiline directive in which each pattern is specified on one line. The pattern is not case sensitive and it accepts simple wildcards such as `*`, `?`, and `[a-z]`. The default patterns that are not accessible from a direct URL are `sys.*`, `dbms_*`, `utl_*`, and `owa_util.*`.

Caution: Setting the *PlsqlExclusionList* directive to **#NONE#** will disable all protection. It is not recommended for an active Web site. Only use this setting for debugging purposes.

If the *PlsqlExclusionList* directive is overridden, the default settings do not apply. In this case, you must add the default list to the list of excluded patterns.

3.3.2 Accessing the PlsqlExclusionList Directive

You can set the *PlsqlExclusionList* directive in the mod_plsql configuration file called `dads.conf`. This configuration file is located in the following directories:

- (UNIX) `ORACLE_HOME/Apache/modplsql/conf/`
- (Windows) `ORACLE_HOME\Apache\modplsql\conf`

Where *`ORACLE_HOME`* is the location of your Oracle Application Server Portal installation.

To ensure the best security for PL/SQL procedures that are granted to PUBLIC, specify the system default settings with the *PlsqlExclusionList* directive in the `dads.conf` file as shown in Example 3–1.

Example 3–1 System Default Settings Specified with the PlsqlExclusionList Directive

```
PlsqlExclusionList sys.*
PlsqlExclusionList dbms_*
PlsqlExclusionList utl_*
PlsqlExclusionList owa_util.*
PlsqlExclusionList portal.wwmon*
```

In addition to the patterns that are specified with this directive, mod_plsql also disallows any URLs that contain special characters such as tabs, newlines, carriage returns, single quotation marks (’), or reverse slashes (\). You cannot change this.

A
Frequently Asked Questions

What is the mod_plsql gateway?

mod_plsql is an Oracle HTTP Server plug-in that communicates with the database by mapping browser requests into database stored procedure calls over a SQL*Net connection. It is generally indicated by a `/pls` virtual path. The mod_plsql gateway provides support for building and deploying PL/SQL-based applications on the Web. PL/SQL stored procedures can retrieve data from database tables and generate HTTP responses containing formatted data and HTML code to display in a Web browser. See the *Oracle Application Server 10g PL/SQL Web Toolkit Reference* for more information.

What is the PL/SQL Web Toolkit?

The PL/SQL Web Toolkit enables you to develop Web applications as PL/SQL procedures stored in an Oracle database server. Packages in the toolkit define procedures, functions, and data types that you can use in your stored procedures. You can use these packages when developing database portlets for OracleAS Portal.

See the *Oracle Application Server 10g PL/SQL Web Toolkit Reference* for more information.

How do I find the version of mod_plsql?

You can determine the version of mod_plsql by executing the `oversioncheck` script on the mod_plsql binary.

On UNIX platforms, issue the following command:

```
ORACLE_HOME/Apache/Apache/bin/oversioncheck ORACLE_
HOME/Apache/modplsql/bin/modplsql.so
```

On Windows platforms, issue the following command:

```
ORACLE_HOME\Apache\Apache\bin\oversioncheck ORACLE_HOME\bin\modplsql.dll
```

How do I find the version of the OWA packages?

1. Use SQL*Plus and connect as any user to the database.
2. Execute the following command:

   ```
   select owa_util.get_version from dual;
   ```

   This should show the version of the OWA packages. For example, 9.0.4.0.1.

If this query fails, you are having a very old version of OWA packages that does not have versioning. It is recommended that you upgrade to a newer version.

How do I install the OWA packages?

See Section 2.2, "Installing Required Packages" for more information.

How do I uninstall the OWA packages?

OWA packages can be uninstalled by performing the following tasks:

1. Navigate to the directory from where the OWA packages were installed. For example:

   ```
   cd ORACLE_HOME/Apache/modplsql
   ```

2. Use SQL*Plus to connect as the owner the OWA packages. This user should be the SYS user, unless you have an old version of the OWA packages.
3. Invoke the script `owadins.sql` to uninstall the OWA packages.

How do I detect and clean up duplicate OWA packages installed in the database?

The following SQL query can be used to determine the location of the OWA packages:

```
SELECT OWNER, OBJECT_TYPE
FROM   DBA_OBJECTS
WHERE  OBJECT_NAME = 'OWA'
```

You will see the following results:

```
SQL>

1  SELECT OWNER, OBJECT_TYPE
2  FROM DBA_OBJECTS
3* WHERE OBJECT_NAME = 'OWA'

OWNER  OBJECT_TYPE
-----  -----------
SYS    PACKAGE
SYS    PACKAGE BODY
PUBLIC SYNONYM
```

If you see more lines than shown above, it means that older OWA packages exist in other schemas which may cause issues for mod_plsql users. In such situations, uninstall all versions of the OWA packages from the database, and reinstall the OWA packages that ship with Oracle Application Server.

I am getting HTTP error codes while accessing PL/SQL procedures via mod_plsql.

mod_plsql logs detailed error messages to the OHS file `*ORACLE_HOME*/Apache/Apache/logs/error_log`. Scan this file to understand the problem. For more information on mod_plsql logging, see "What kind of logging facilities are available in mod_plsql?".

All my PL/SQL procedures return a "Document contains no data" error in Netscape, or a blank page in Internet Explorer.

This problem could occur if you have duplicate OWA Packages installed in the database. See "How do I detect and clean up duplicate OWA packages installed in the database?" for more information.

I have a performant PL/SQL procedure, but some of my HTTP requests via mod_plsql take more than 15 seconds.

The most common reason for this problem is that the Oracle Application Server middle-tier character set does not match that of the back-end database, and *HTTP KeepAlive* is enabled in Oracle HTTP Server. This kind of misconfiguration causes an invalid *Content-Length* to be sent back to the browser, causing the browser to detect the end of the response stream only when the *KeepAliveTimeout* interval causes the stream to be closed. To solve the problem, ensure that the *PlsqlNLSLanguage* parameter in the DAD matches that of the database.

Can I use mod_plsql to run applications on my own database?

Yes. But before you can run your applications, you need to install the OWA packages into your database. See Section 2.2, "Installing Required Packages".

How do I configure mod_plsql?

Please refer to the *Oracle HTTP Server Administrator's Guide*.

How do I create a DAD through the Oracle Enterprise Manager UI?

Refer to Oracle Enterprise Manager documentation for information on creating DADs.

How do I troubleshoot problems with DAD configuration done through the Oracle Enterprise Manager UI?

The DAD configuration tool outputs all log details from using the tool, to a log file `ORACLE_HOME/Apache/modpslql/logs/log.xml`.

Can I make manual edits to the DAD configuration file?

Yes. But, you should run the following command after the update:

```
ORACLE_HOME/dcm/bin/dcmctl updateConfig -ct ohs
```

This will update the Oracle Enterprise Manager repository with the configuration change and will obscure the DAD password.

What authentication modes are available in mod_plsql?

See Chapter 3, "Securing Application Database Access Through mod_plsql".

What is the mod_plsql Cleanup Thread?

mod_plsql starts a thread in each httpd process. The job of this thread is to clean up idle database sessions and the file system cache. This thread is called the **Cleanup Thread**.

What kind of database connection pooling is present in mod_plsql?

Please refer to the *Oracle Application Server 10g Performance Guide*.

How does mod_plsql clean up database sessions?

mod_plsql cleans up unused database sessions based on the configuration setting of *PlsqlIdleSessionCleanupInterval*. Besides this, the configuration directive *PlsqlMaxRequestsPerSession* governs how many requests will be serviced from a pooled database session. Finally, database sessions are closed when httpd processes are shut down.

What happens when pooled database connections exist in mod_plsql and the database is restarted?

When the database connection is severed, the first request which attempts to execute a PL/SQL procedure using the severed connection will fail. Subsequent requests will reestablish a database session and start functioning normally. The number of failures will be directly proportional to the number of pooled database sessions. Future versions of mod_plsql will detect dead connections automatically.

Note: If the database is not restarted within the time interval of *PlsqlIdleSessionCleanupInterval*, then the cleanup thread will clean up the severed sessions, and no errors will be seen by end-users.

How does mod_plsql clean up cached content in the file system?

The cleanup thread scans the Portal file system cache based on the configuration of *PlsqlCacheCleanupTime*. The default cleanup time is everyday at 11 P.M. local time.

See Also: *Oracle Application Server 10g Performance Guide*

Can I invoke mod_plsql without a "/pls" prefix in the URL?

Yes. Since mod_plsql uses the OHS' *Location* directive, you can configure any virtual path to be serviced by mod_plsql.

How can I improve PL/SQL and mod_plsql performance?

Please refer to the *Oracle Application Server 10g Performance Guide*.

What kind of logging facilities are available in mod_plsql?

- By default, mod_plsql logs alerts/warnings/errors to the OHS `error_log` file *`ORACLE_HOME`*`/Apache/Apache/logs/error_log`. The amount of information logged by mod_plsql is controlled by the setting of OHS' *LogLevel* parameter in `httpd.conf`. By default, this is configured to *warn*.
- You can also enable performance logging for mod_plsql on a per-request basis as follows:

  1. Edit *`ORACLE_HOME`*`/Apache/Apache/conf/httpd.conf` and set *LogLevel* to *info* (default is *warn*).
  2. Issue the following command to update DCM with the configuration change:

     ```
     ORACLE_HOME/dcm/bin/dcmctl updateConfig -ct ohs
     ```

  3. Restart OHS using the following command:

     ```
     ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
     ```

  4. Issue some URLs to mod_plsql and verify that the file *`ORACLE_HOME`*`/Apache/Apache/logs/error_log` starts showing entries as follows:

     ```
     [Tue Apr 01 14:54:49 2003] [info] mod_plsql: [perf] 130.35.92.145
     /pls/app/htp.p status=200 user=scott reqTime=21ms connSU=(null),0ms
     connRO=(null),0ms connNSSO=HIT,1ms procTime=17ms sessionTidyTime=0ms
     cache=(null) cookie=(null),0ms pageCalls=0,0ms bytes=5 describe=No,0ms
     streamTime=0ms pid=175 sessFile=(null) userFile=834\0855
     sysFile=470\5949 cacheLevel=(null) cacheTime=0ms dbProcTime=15ms
     id=1049237685:130.35.92.145:373:1 spid=(null) qs=(null)
     requestTrace=(null) cookieLen=0 cookieValue=(null) reqUserTime=16ms
     assertUser=(null) subid=(null) authLevel=(null) oraError=0
     ```

- If you are a Portal customer, you can additionally choose to enable Portal session cookie logging by adding the following configuration parameter to your Portal DAD:

  ```
  PlsqlInfoLogging InfoDebug
  ```

  This is in addition to the previous setting of *LogLevel* to *info* in `httpd.conf`.

- Finally, you can enable debug logging in mod_plsql. This is the highest level of logging and is not recommended for active sites.

Caution: This mode of logging should be enabled only on the request of Oracle Support.

In this mode, debug messages are logged to Oracle HTTP Server's `error_log` file and additional mod_plsql specific logs are created under *`ORACLE_HOME`*`/Apache/modplsql/logs`. Log location is configurable using the *PlsqlLogDirectory* directive in *`ORACLE_HOME`*`/Apache/modplsql/conf/plsql.conf`. To enable debug level logging:

1. Edit *`ORACLE_HOME`*`/Apache/modplsql/conf/plsql.conf` and set *PlsqlLogEnable* to **On** (default is **Off**).
2. Issue the following command to update DCM with the configuration change:

```
ORACLE_HOME/dcm/bin/dcmctl updateConfig -ct ohs
```

3. Restart OHS using the following command:

```
ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
```

What kind of DMS metrics are available for mod_plsql?

Please refer to the *Oracle Application Server 10g Performance Guide*.

What considerations should I have in mod_plsql for High Availability?

For high availability, mod_plsql based applications should be aware of the following things:

- The mod_plsql configuration parameter *PlsqlDatabaseConnectString* should use a connect string format of *NetServiceNameFormat* so that name resolution happens via an LDAP lookup of Oracle Internet Directory. This allows you to configure the database *host:port:service_name* information in a central repository, which makes it easier to add or remove RAC nodes when required.
- mod_plsql does not automatically detect dead database connections. In case a back end database goes down, the initial few requests fail. See "What happens

when pooled database connections exist in mod_plsql and the database is restarted?" for more information.

What considerations should I have in mod_plsql when the database is separated by a firewall?

If a firewall exists between the Oracle Application Server middle-tier running mod_plsql, and the back end database, the idle session cleanup interval in mod_plsql should be configured lower than the idle session cleanup interval of the firewall. This ensures that the firewall never closes a connection established by mod_plsql.

> **Note:** mod_plsql idle session cleanup interval can be configured using the parameter *PlsqlIdleSessionCleanupInterval* in `ORACLE_HOME/Apache/modplsql/conf/plsql.conf`. The default value is 15 minutes.

How do I assert a different hostname, port, or request_protocol to the PL/SQL application?

- In situations where your OHS instance is front-ended by Web Cache or a Load Balancing Router, there is a need to assert the hostname and port for the site to be that of the Web Cache or the LBR. In such situations, it is recommended that you use the OHS configuration directives *ServerName* and *Port* to do the assertion. If for some reason, you do not wish to assert the hostname and port at the OHS level, you can use the mod_plsql configuration directive *PlsqlCGIEnvironmentList* to assert a different hostname and port to only the PL/SQL applications running under mod_plsql. For example:
  - `PlsqlCGIEnvironmentList  SERVER_NAME=lbr.us.oracle.com`

    Consider using OHS' *ServerName* directive in `httpd.conf` instead.
  - `PlsqlCGIEnvironmentList  SERVER_PORT=9999`

    Consider using OHS' *Port* directive in `httpd.conf` instead.
  - `PlsqlCGIEnvironmentList  HTTP_HOST=myservername.us.oracle.com:9999`

    Combination of `SERVER_NAME:SERVER_PORT`.
- Similarly, in cases where your site is accessed externally as an SSL, but is internally running in non-SSL mode (with an SSL accelerator in between), you

might want to assert the REQUEST_PROTOCOL as *HTTPS* so that the PL/SQL application generates SSL links instead of non-SSL links. For example:

```
PlsqlCGIEnvironmentList  REQUEST_PROTOCOL=https
```

How do I disable access to procedure names which have a specific pattern?

Please refer to the description of *PlsqlExclusionList* in Section 3.3.1, "Using the PlsqlExclusionList Directive in mod_plsql".

I see the error "HTTP-503 ORA-12154" in the file ORACLE_ HOME/Apache/Apache/conf/error_log. What does this mean?

This error means that mod_plsql is unable to connect to the database.

Ensure that:

1. The database is up and running.
2. The username and password information in the DAD is correct.
3. The Oracle Application Server middle-tier is able to connect to the database using the *PlsqlDatabaseConnectString* parameter in the DAD.

In most situations, the problem occurs because SQL*Net is not able to resolve the connect string parameter using the configuration information under `ORACLE_HOME/network/admin`.

- For entries configured with *TNSFormat* or *NetServiceNameFormat*, validate the connect string information by using `tnsping dad_connect_string`. For example:

  ```
  tnsping "cn=iasdb,cn=oraclecontext"
  ```

  or

  ```
  tnsping iasdb.us.oracle.com
  ```

- For entries configured with *SIDFormat* and *ServiceNameFormat*, ensure that the hostname, port, and SID/service_name information match for the database listener. After verifying this, confirm that SQL*Plus can connect to the database using the DAD username, password, and connect string.

If this does not work, refer to the Oracle SQL*Net documentation on how to troubleshoot this further.

Note: In 10*g* (9.0.4), the default connect string parameter in the DAD is configured to get resolved via an LDAP lookup in Oracle Internet Directory. If you make changes to `ldap.ora`, you will need to restart Oracle HTTP Server for the changes to be accessible to mod_plsql.

Index