Oracle9i Application Server

mod_plsql User’s Guide

Release 2 (9.0.2)

February 2002

Part No. A90855-01

mod_plsql User’s Guide, Release 2 (9.0.2)

Part No. A90855-01

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

Contents

Send Us Your Comments

mod_plsql User's Guide, Release 2 (9.0.2)

Part No. A90855-01

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

- Electronic mail: iasdocs_us@oracle.com

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

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

Preface

This manual describes how to install, configure, and maintain mod_plsql for Oracle9*i* Application Server v 9.0.2. It contains the following chapters:

Chapter 1 - Explains how to use mod_plsql.

Chapter 2 - Provides an overview of the mod_plsql and its features.

Related Oracle Documents

For more information, see the following manuals:

Titles within the Oracle9*i* Application Server set	**Part Number**	**Containing Information about...**
Oracle HTTP Server Administration Guide	A92173-01	Oracle HTTP Server Modules - Administration using the command line tools and manually editing configuration files (DAD parameters, http.conf etc.)
Oracle9i Application Server Security Guide	A90146-01	Securing Database Access through mod_plsql
Oracle9i Application Server Performance Guide	A95102-01	Performance and tuning material and caching
Oracle9i Application Server Administrator's Guide	A92171-01	Administering Oracle9*i* Application Server through the Oracle Enterprise Manager Console (DAD Configuration)

Titles within the Oracle9*i* Application Server set	Part Number	Containing Information about...
Oracle9i Application Server: Migrating from Oracle9iAS Release 1 (1.0.2.2.x) to Release 2 (9.0.2)	A96157-01	Parameters that have changed and tools used to migrate DADs
PL/SQL Web Toolkit Reference	A90101-01	OWA package information
Oracle9i Application Server Installation Guide	Solaris: A90215-01 NT: A90216-01	Installation for the Oracle9*i* Application Server
Oracle9i Application Server: Migrating from Oracle Application Server	A95108-01	Migrating from previous versions (Oracle Application Server)
Oracle9i Application Server Concepts	A95926-01	Overview of the Oracle9*i* Application Server

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://technet.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`

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.

Convention	Meaning
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Documentation Accessibility

1 Using mod_plsql

1.1 Oracle Database Requirements

The following are the recommended and minimum requirements for the Oracle database when using mod_plsql:

- Oracle8*i* or higher

> **Note:** mod_plsql requires the Oracle 9.0.1.2 client libraries to be installed in the same Oracle Home as mod_plsql. If these libraries are installed, you can still run mod_plsql against remote Oracle 8*i* or above databases.

- For 8i databases, verify you are on the correct path set.
- For Solaris, the database must have the following patch sets:
  - 8.1.6.3 or higher for 8.1.6 databases
  - 8.1.7.2 or higher for 8.1.7 databases
- For Windows NT, you need:
  - 8.1.6.3.1 or higher for 8.1.6 databases
  - 8.1.7.1.1 or higher for 8.1.7 databases

1.2 Before you begin

Before you install mod_plsql using the Oracle9*i* Application Server Oracle Universal Installer, satisfy the following requirements:

- You must have a SYS user password on the database where you plan to load PL/SQL Web Toolkit packages required by mod_plsql.
- The database to which you plan to connect mod_plsql 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.

1.3 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. If you think other OWA packages exist in SYS, run the following query to produce a list of all instances of the OWA packages

   ```
   'select object_name, owner from all_objects where object_name like HTP%';
   ```

   If multiple instances are detected in other schemas, deinstall other versions before installing the OWA packages.
4. At a SQL prompt, run the following command:

   ```
   @owaload.sql log_file
   ```

Table 1–1 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.

7. After the install, check the version of the OWA packages by running "Select owa_util.get_version". Confirm that the version shown is 9.0.2.1.0 or above.

1.3.1 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.
3. Install the new PL/SQL Web Toolkit as per "Installing Required Packages" on page 1-2.

1.4 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.

1.4.1 Access the DAD Configuration pages through OEM

1. Enter the following URL in the web browser:
   http://<hostname>:<port_number>

   Note: 1810 is the default port.

2. Enter the Oracle9*i* Application Server administrator username and password. The default username for administrator user is ias_admin. The default password is defined during the installation of Oracle9*i*AS.
3. Click **OK**.
4. Select Oracle9*i* 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 select the DAD you are interested in and click **Edit**.

1.4.2 Access the DAD Configuration pages through Portal

1. Log on to Portal.
2. Click the **Builder** icon.
3. Access the **Administrator** 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 select the DAD you are interested in and click **Edit**.

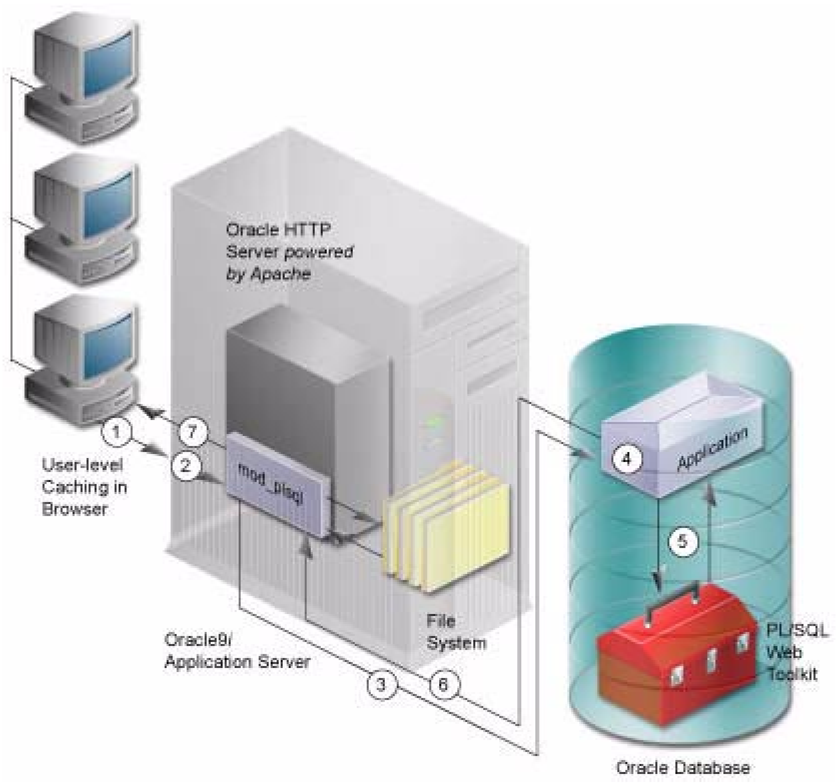
2 mod_plsql Overview

Oracle9*i* 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.

2.1 Processing Client Requests

Mod_plsql is an Apache plug-in that communicates with the database. It maps browser requests into database stored procedure calls over a SQL*Net connection. It is generally indicated by a */pls* virtual path.

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


Oracle HTTP Server *powered by Apache*
mod_plsql
User-level Caching in Browser
File System
Oracle9*i* Application Server
Application
PL/SQL Web Toolkit
Oracle Database
1
2
3
4
5
6
7

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.

2.2 Database Access Descriptors

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.

2.3 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 2–1 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.
DAD location	A virtual path to handle PL/SQL requests that you have configured in the WEb server.
! character (optional)	Indicates to use the flexible parameter passing scheme. See"Flexible Parameter Passing" on page 2-8 for more information.

Table 2–1 Invoking mod_plsql Parameters

Parameter	Description
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 2-5 for more information.

Example 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 `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 DAD Configuration pages) displays.

2.3.1 POST, GET and HEAD Methods

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

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"

2.4 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.

2.5 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 "Parameter Passing by Name (Overloaded parameters)" on page 2-6 for more information.

- Flexible parameter passing

  Procedures are prefixed by a ! character. See "Flexible Parameter Passing" on page 2-8 for more information.

- Large (up to 32K) parameters passing

  See "Large Parameter Passing" on page 2-9 for more information.

2.5.1 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, 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
```

2.5.1.1 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: 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 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.

2.5.2 Flexible Parameter Passing

mod_plsql 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 mod_plsql. The four parameter interface is supported for compatibility.

2.5.2.1 Two parameter interface

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

Table 2–2 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.
array_type (required)	The values from the query string (indexed from 1) in the order submitted.

Example: If you send the following URL:

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

The exclamation mark prefix (!) instructs mod_plsql 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’)
```

2.5.2.2 Four parameter interface

The four parameter interface is supported for compatibility.

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

Table 2–3 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.
value_array	The values from the query string (indexed from 1) in the order submitted.
reserved	Not used. It is reserved for future use.
array_type (required)	Any PL/SQL index-by table of varchar2 type (e.g., owa.vc_arr).

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

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

The exclamation mark prefix (!) instructs 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’);
values_array ==> (‘a’, ‘b’, ‘c’)
reserved ==> ()
```

2.5.3 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 2-8), 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.

2.6 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.

2.6.1 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 ;
```

2.6.1.1 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.

2.6.1.2 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'.

2.6.1.3 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.

2.6.1.4 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.

2.6.2 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
);
```

2.6.3 Parameters for Document Upload/Downloading

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

PlsqlDocumentTablename

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

Syntax:

```
PlsqlDocumentTablename  [document_table_name]
```

Examples:

```
PlsqlDocumentTablename  my_documents
```

or,

```
PlsqlDocumentTablename  scott.my_document_table
```

2.6.3.1 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://neon/pls/mydad/docs/myfile.htm
```

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

Syntax:

```
PlsqlDocumentPath  [document_access_path_name]
```

2.6.3.2 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 "File Download" on page 2-16.

Syntax:

```
PlsqlDocumentProcedure  [document_access_procedure_name]
```

Examples:

```
PlsqlDocumentProcedure  my_access_procedure
```
or,

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

2.6.3.3 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 *.

Examples:

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

2.6.4 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).

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`.

2.6.5 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 "File Upload" on page 2-14 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.

2.6.6 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).
```

2.6.7 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: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.

2.6.8 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.get_length(myblob));
owa_util.http_header_close;
```

c. Initiate Direct BLOB download

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

The structure of the mytable table:

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

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

2.7 Path Aliasing (Direct Access URLs)

Path Aliasing enables applications using mod_plsql to provide direct reference to its objects using simple URLs. This lets you 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 mod_plsql that this request is for document access.

```
protocol://hostname[:port]/DAD
Location/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 mod_plsql encounters an incoming URL with the keyword entered in the **Path Alias** field, it invokes the procedure entered in the **Path Alias Procedure** field.

For example, if the URL below is the incoming URL and the Path Alias is set to `myalias`, mod_plsql invokes the **Path Alias Procedure. This passes** everything after the keyword `myalias` to the invoked procedure.

```
http://www.acme.com:9000/pls/mydad/myalias/foo/bar/foobar
```

Applications that use path aliasing must implement the **Path Alias Procedure**. The procedure receives the rest of the URL (`foo/bar/foobar`) after the keyword, `myalias`, as a single parameter. It is responsible for 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.

2.8 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. The following are the available CGI Environment Variables:

Table 2–4 CGI Environment Variables

CGI Environment Variables	
AUTHORIZATION	PlsqlDocumentTablename
DAD_NAME	REMOTE_ADDR
DOC_ACCESS_PATH	REMOTE_HOST
HTTP_ACCEPT	REMOTE_USER
HTTP_ACCEPT_CHARSET	REQUEST_CHARSET (refer to "REQUEST_ CHARSET CGI environment variable" on page 2-21)

Table 2–4 CGI Environment Variables

CGI Environment Variables	
HTTP_ACCEPT_LANGUAGE	REQUEST_IANA_CHARSET
HTTP_COOKIE	REQUEST_METHOD
HTTP_HOST	REQUEST_PROTOCOL
HTTP_PRAGMA	SCRIPT_NAME
HTTP_REFERER	SCRIPT_PREFIX
HTTP_USER_AGENT	SERVER_NAME
PATH_ALIAS	SERVER_PORT
PATH_INFO	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.

2.8.1 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 "Common Gateway Interface (CGI) Environment Variables" on page 2-19), 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 Administration 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 Apache, 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:

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 2:

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".

2.8.2 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.

2.8.2.1 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');
```

2.8.2.2 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.

2.9 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.

Index