Oracle® Application Express

API Reference

Release 5.1

E64915-04

June 2017

ORACLE

Oracle Application Express API Reference, Release 5.1

E64915-04

Primary Author: Harish Konakondla

Contributors: Terri Jennings, Christina Cho, Hilary Farrell , Joel Kallman, Sharon Kennedy, Christian Neumueller, Anthony Raynor, Marc Sewtz, John Snyders, Jason Straub, Drue Swadener, Vladislav Unarov, Patrick Wolf

Contents

27 APEX_UI_DEFAULT_UPDATE

28 APEX_UTIL

Preface

Oracle Application Express API Reference describes the Application Programming Interfaces, referred to as APIs, available when programming in the Oracle Application Express environment. To utilize these APIs, such as APEX_JSON, when not developing with Oracle Application Express, you need to install Oracle Application Express into the database.

Audience

Oracle Application Express API Reference is intended for application developers who are building database-centric web applications using Oracle Application Express. The guide describes the APIs available when programming in the Oracle Application Express environment.

To use this guide, you need to have a general understanding of relational database concepts and an understanding of the operating system environment under which you are running Oracle Application Express.

See Also:

Oracle Application Express App Builder User's Guide

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Application Express Release Notes*
- *Oracle Application Express Installation Guide*
- *Oracle Application Express App Builder User's Guide*
- *Oracle Application Express Administration Guide*
- *Oracle Application Express Application Migration Guide*
- *Oracle Application Express SQL Workshop Guide*
- *Oracle Application Express End User Guide*
- *Oracle Database Concepts*
- *Oracle Database Advanced Application Developer's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Language Reference*
- *SQL*Plus User's Guide and Reference*
- *Oracle Database PL/SQL Language Reference*

Conventions

For a description of PL/SQL subprogram conventions, refer to the *Oracle Database PL/SQL Language Reference*. This document contains the following information:

- Specifying subprogram parameter modes
- Specifying default values for subprogram parameters
- Overloading PL/SQL subprogram Names

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release

This preface contains:

Changes in Oracle Application Express Release 5.1

The following are changes in *Oracle Application Express API Reference* for Oracle Application Express release 5.1.

New Features

The following features are new in this release:

- APEX_UTIL (Updates)

  SET_APP_BUILD_STATUS Procedure (New) - This procedure sets the build status of the specified application.

  GET_SINCE Function (New) - This function returns the relative date in words (for example, 2 days from now, 30 minutes ago).

  SET_APPLICATION_STATUS (New) - This procedure changes the status of the application.

  GET_SUPPORTING_OBJECT_SCRIPT Function (New) - This function gets supporting object scripts defined in an application.

  GET_SUPPORTING_OBJECT_SCRIPT Procedure (New) - This procedure gets supporting object scripts and outputs to `sys.dbms_output` buffer or download as a file.

- JavaScript APIs - apex.server namespace (Updates)

  apex.server.loadScript (New) - API used to asynchronously load other JavaScript libraries that has Asynchronous Module Definition (AMD), and if `RequireJS` library is already loaded on the page.

- APEX_APPLICATION_INSTALL (Updates)

  SET_KEEP_SESSIONS Procedure (New) -This procedure preserves sessions associated with the application on upgrades.

GET_KEEP_SESSIONS Function (New) - This function finds out if sessions and session state will be preserved or deleted on upgrades.

- APEX_INSTANCE_ADMIN (Updates)

  Available Parameter Values (Updates) - Added a parameter `KEEP_SESSIONS_ON_UPGRADE`

- APEX_JSON (Updates)

  GET_CLOB Function (New) - This function returns `clob` member value. This function auto-converts `varchar2`, boolean and number values.

  Constants and Data Types (Updates) - Added `c_clob` constant used for the parser interface.

  WRITE Procedures (New) - Procedures to write array attribute of type `varchar2` and an array attribute of type number.

- APEX_STRING (New)

  This is a new package that provides utilities for `varchar2`, `clob`, `apex_t_varchar2`, and `apex_t_number` types.

- APEX_WEB_SERVICE (Updates)

  OAUTH_AUTHENTICATE Function (New) - This function performs OAUTH autentication and requests an OAuth access token.

  OAUTH_GET_LAST_TOKEN Function (New) - This function returns the OAuth access token received in the last `OAUTH_AUTHENTICATE` call.

  OAUTH_SET_TOKEN Function (New) - This function sets the OAuth Access token to be used in subsequence `MAKE_REST_REQUEST` calls.

- APEX_DEBUG (Updates)

  ENABLE_DBMS_OUTPUT Procedure (New) - This procedure writes all debug logs via `dbms_output`.

  DISABLE_DBMS_OUTPUT Procedure (New) - This procedure stops writing all debug logs also via `dbms_output`.

- APEX_ITEM (Updates)

  SWITCH Function (New) - This function dynamically generates flip toggle item.

- REST Administration Interface API (New)

  The REST Administration API enables Oracle Application Express instance administrator to perform administrative functions in an Application Express instance over REST and HTTP protocols. This is particularly useful for machine-to-machine communication when SQL*Net connections are not possible, for instance in cloud environments.

- CSS calendar component (Updates)

  Enables CSS Calendar to display only week or day views when `"Show Time"` equals `YES`.

- APEX_JAVASCRIPT (Updates)

ADD_LIBRARY Procedure (Updates) - Added new parameters `p_requirejs_module`, `p_requirejs_js_expression` and `p_requirejs_required`.

- APEX_PLUGIN_UTIL (Updates)

  GET_DATA2 Function Signature 1 (New) - Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column.

  GET_DATA2 Function Signature 2 (New) - Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column.

- APEX_SESSION (NEW)

  The package enables you to configure Application Express sessions.

- APEX_IR (Updates)

  GET_REPORT Function (New) - This function returns an interactive report runtime query.

Deprecated and Desupported Features

See "Deprecated Features" and "Desupported Features" in *Oracle Application Express Release Notes*.

1

APEX_APPLICATION

The APEX_APPLICATION package is a PL/SQL package that implements the Oracle Application Express rendering engine. You can use this package to take advantage of many global variables. "Global Variables Available in APEX_APPLICATION" describes the global variables available in the APEX_APPLICATION package.

Note:

"Global Variables (page 1-1)"

1.1 Global Variables

Table 1-1 Global Variables Available in APEX_APPLICATION

Global Variable	Description
G_USER	Specifies the currently logged in user.
G_FLOW_ID	Specifies the ID of the currently running application.
G_FLOW_STEP_ID	Specifies the ID of the currently running page.
G_FLOW_OWNER	Specifies the schema to parse for the currently running application.
G_REQUEST	Specifies the value of the request variable most recently passed to or set within the show or accept modules.
G_BROWSER_LANGUAGE	Refers to the web browser's current language preference.
G_DEBUG	Refers to whether debugging is currently switched on or off. Valid values for the DEBUG flag are 'Yes' or 'No'. Turning debug on shows details about application processing.

Table 1-1 (Cont.) Global Variables Available in APEX_APPLICATION

Global Variable	Description
G_HOME_LINK	Refers to the home page of an application. The Application Express engine redirects to this location if no page is given and if no alternative page is dictated by the authentication scheme's logic.
G_LOGIN_URL	Used to display a link to a login page for users that are not currently logged in.
G_IMAGE_PREFIX	Refers to the virtual path the web server uses to point to the images directory distributed with Oracle Application Express.
G_FLOW_SCHEMA_OWNER	Refers to the owner of the Application Express schema.
G_PRINTER_FRIENDLY	Refers to whether the Application Express engine is running in print view mode. This setting can be referenced in conditions to eliminate elements not desired in a printed document from a page.
G_PROXY_SERVER	Refers to the application attribute 'Proxy Server'.
G_SYSDATE	Refers to the current date on the database server. this uses the DATE DATATYPE.
G_PUBLIC_USER	Refers to the Oracle schema used to connect to the database through the database access descriptor (DAD).
G_GLOBAL_NOTIFICATION	Specifies the application's global notification attribute.
G_X01, ... G_X10	Specifies the values of the X01, ... X10 variables most recently passed to or set within the show or accept modules. You typically use these variables in On Demand AJAX processes.

1.2 Referencing Arrays

Items are typically HTML form elements such as text fields, select lists, and check boxes. When you create a new form item using a wizard, the wizard uses a standard naming format. The naming format provides a handle so you can retrieve the value of the item later on.

To create your own items, you can access them after a page is submitted by referencing APEX_APPLICATION.G_F01 to APEX_APPLICATION.G_F50 arrays. You can create your own HTML form fields by providing the input parameters using the format F01, F02, F03 and so on. You can create up to 50 input parameters ranging from F01 to F50, for example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="some value">

<TEXTAREA NAME="F02" ROWS=4 COLS=90 WRAP="VIRTUAL">this is the example of a text
area.</TEXTAREA>

<SELECT NAME="F03" SIZE="1">
<OPTION VALUE="abc">abc
<OPTION VALUE="123">123
</SELECT>
```

Because the F01 to F50 input items are declared as PL/SQL arrays, you can have multiple items named the same value. For example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 1">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 2">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 3">
```

Note that following PL/SQL code produces the same HTML as show in the previous example.

```
FOR i IN 1..3 LOOP
APEX_ITEM.TEXT(P_IDX        => 1,
   p_value      =>'array element '||i ,
   p_size       =>32,
   p_maxlength  =>32);
END LOOP;
```

1.3 Referencing Values Within an On Submit Process

You can reference the values posted by an HTML form using the PL/SQL variable APEX_APPLICATION.G_F01 to APEX_APPLICATION.G_F50. Because this element is an array, you can reference values directly, for example:

```
FOR i IN 1..APEX_APPLICATION.G_F01.COUNT LOOP
    htp.p('element '||I||' has a value of '||APEX_APPLICATION.G_F01(i));
END LOOP;
```

Note that check boxes displayed using APEX_ITEM.CHECKBOX only contain values in the APEX_APPLICATION arrays for those rows which are checked. Unlike other items (TEXT, TEXTAREA, and DATE_POPUP) which can contain an entry in the corresponding APEX_APPLICATION array for every row submitted, a check box only has an entry in the APEX_APPLICATION array if it is selected.

1.4 Converting an Array to a Single Value

You can also use Oracle Application Express public utility functions to convert an array into a single value. The resulting string value is a colon-separated list of the array element values. For example:

```
htp.p(APEX_UTIL.TABLE_TO_STRING(APEX_APPLICATION.G_F01));
```

This function enables you to reference G_F01 to G_F50 values in an application process that performs actions on data. The following sample process demonstrates how values are inserted into a table:

```
INSERT INTO my_table (my_column) VALUES
APEX_UTIL.TABLE_TO_STRING(APEX_APPLICATION.G_F01)
```

1.5 HELP Procedure

This function outputs page and item level help text as formatted HTML. You can also use it to customize how help information is displayed in your application.

Syntax

```
APEX_APPLICATION.HELP (
    p_request        IN VARCHAR2 DEFAULT NULL,
    p_flow_id        IN VARCHAR2 DEFAULT NULL,
    p_flow_step_id   IN VARCHAR2 DEFAULT NULL,
```

```
p_show_item_help IN VARCHAR2 DEFAULT 'YES',
p_show_regions   IN VARCHAR2 DEFAULT 'YES',
p_before_page_html     IN VARCHAR2 DEFAULT '<p>',
p_after_page_html      IN VARCHAR2 DEFAULT NULL,
p_before_region_html   IN VARCHAR2 DEFAULT NULL,
p_after_region_html    IN VARCHAR2 DEFAULT '</td></tr></table></p>',
p_before_prompt_html   IN VARCHAR2 DEFAULT '<p><b>',
p_after_prompt_html    IN VARCHAR2 DEFAULT '</b></p>: ',
p_before_item_html     IN VARCHAR2 DEFAULT NULL,
p_after_item_html      IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1-2 (page 1-4) describes the parameters available in the HELP procedure.

Table 1-2 HELP Parameters

Parameter	Description
p_request	Not used.
p_flow_id	The application ID that contains the page or item level help you want to output.
p_flow_step_id	The page ID that contains the page or item level help you want to display.
p_show_item_help	Flag to determine if item level help is output. If this parameter is supplied, the value must be either 'YES' or 'NO', if not the default value is 'YES'.
p_show_regions	Flag to determine if region headers are output (for regions containing page items). If this parameter is supplied, the value must be either 'YES' or 'NO', if not the default value is 'YES'.
p_before_page_html	Use this parameter to include HTML between the page level help text and item level help text.
p_after_page_html	Use this parameter to include HTML at the bottom of the output, after all other help.
p_before_region_html	Use this parameter to include HTML before every region section. Note this parameter is ignored if p_show_regions is set to 'NO'.
p_after_region_html	Use this parameter to include HTML after every region section. Note this parameter is ignored if p_show_regions is set to 'NO'.
p_before_prompt_html	Use this parameter to include HTML before every item label for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.
p_after_prompt_html	Use this parameter to include HTML after every item label for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.
p_before_item_html	Use this parameter to include HTML before every item help text for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.

Table 1-2 (Cont.) HELP Parameters

Parameter	Description
p_after_item_html	Use this parameter to include HTML after every item help text for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.

Example

The following example shows how to use the APEX_APPLICATION.HELP procedure to customize how help information is displayed.

In this example, the p_flow_step_id parameter is set to :REQUEST, which means that a page ID specified in the REQUEST section of the URL controls which page's help information to display (see note after example for full details on how this can be achieved).

Also, the help display has been customized so that the region sub-header now has a different color (through the p_before_region_html parameter) and also the ':' has been removed that appeared by default after every item prompt (through the p_after_prompt_html parameter).

```
APEX_APPLICATION.HELP(
    p_flow_id => :APP_ID,
    p_flow_step_id => :REQUEST,
    p_before_region_html => '<p><br/><table bgcolor="#A3BED8"
width="100%"><tr><td><b>',
    p_after_prompt_html  => '</b></p>  ');
```

To implement this type of call in your application, you can do the following:

1. Create a page that will be your application help page.
2. Create a region of type 'PL/SQL Dynamic Content' and add the APEX_APPLICATION.HELP call as PL/SQL Source.
3. Then you can add a 'Navigation Bar' link to this page, ensuring that the REQUEST value set in the link is &APP_PAGE_ID.

1.6 STOP_APEX_ENGINE Procedure

This procedure signals the Application Express engine to stop further processing and immediately exit to avoid adding additional HTML code to the HTTP buffer.

Note:

This procedure raises the exception apex_application.e_stop_apex_engine internally. You must raise that exception again, if you use a WHEN OTHERS exception handler.

Syntax

```
APEX_APPLICATION.STOP_APEX_ENGINE
```

Parameters

None

Example 1

This example tells the browser to redirect to `http://apex.oracle.com/` and immediately stops further processing.

```
owa_util.redirect_url('http://apex.oracle.com');
apex_application.stop_apex_engine;
```

Example 2

This example also tells the browser to redirect to `http://apex.oracle.com/` and immediately stops further processing. But, this time the code also contains a WHEN OTHERS exception handler which deals with the `apex_application.e_stop_apex_engine` used by `apex_application.stop_apex_engine`.

```
begin
    ... code which can raise an exception ...
    owa_util.redirect_url('http://apex.oracle.com');
    apex_application.stop_apex_engine;
exception
    when apex_application.e_stop_apex_engine then
        raise; -- raise again the stop Application Express engine exception
    when others then
        ...; -- code to handle the exception
end;
```

2

APEX_APPLICATION_INSTALL

The `APEX_APPLICATION_INSTALL` package provides many methods to modify application attributes during the Application Express application installation process.

SET_WORKSPACE_ID Procedure (page 2-17)

SET_WORKSPACE_Procedure (page 2-18)

2.1 Package Overview

Oracle Application Express provides two ways to import an application into an Application Express instance:

1. Upload and installation of an application export file by using the web interface of Application Express.
2. Execution of the application export file as a SQL script, typically in the command-line utility SQL*Plus.

Using the file upload capability of the web interface of Application Express, developers can import an application with a different application ID, different workspace ID and different parsing schema. But when importing an application by using a command-line tool like SQL*Plus, none of these attributes (application ID, workspace ID, parsing schema) can be changed without directly modifying the application export file.

To view the install log, enter the following from the command-line tool, so the server outputs are displayed:

```
set serveroutput on unlimited
```

As more and more Application Express customers create applications which are meant to be deployed by using command-line utilities or by using a non-web-based installer, they are faced with this challenge of how to import their application into an arbitrary workspace on any Application Express instance.

Another common scenario is in a training class when installing an application into 50 different workspaces that all use the same application export file. Today, customers work around this by adding their own global variables to an application export file and then varying the values of these globals at installation time. However, this manual modification of the application export file (usually done with a post-export sed or awk script) should not be necessary.

Application Express 4.0 and higher includes the APEX_APPLICATION_INSTALL API. This PL/SQL API provides many methods to set application attributes during the Application Express application installation process. All export files in Application Express 4.0 and higher contain references to the values set by the APEX_APPLICATION_INSTALL API. However, the methods in this API is only used to override the default application installation behavior.

2.2 Attributes Manipulated by APEX_APPLICATION_INSTALL

The table below lists the attributes that can be set by functions in this API.

Table 2-1 Attributes Manipulated by the APEX_APPLICATION_INSTALL API

Attribute	Description
`Workspace ID`	Workspace ID of the imported application. See GET_WORKSPACE_ID Function (page 2-11), SET_WORKSPACE_ID Procedure (page 2-17).

Table 2-1 (Cont.) Attributes Manipulated by the APEX_APPLICATION_INSTALL API

Attribute	Description
`Application ID`	Application ID of the imported application. See GENERATE_APPLICATION_ID Procedure (page 2-5), GET_APPLICATION_ID Function (page 2-7), SET_APPLICATION_ID Procedure (page 2-12).
`Offset`	Offset value used during application import. See GENERATE_OFFSET Procedure (page 2-6), GET_OFFSET Function (page 2-9), SET_OFFSET Procedure (page 2-15).
`Schema`	The parsing schema ("owner") of the imported application. See GET_SCHEMA Function (page 2-10), SET_SCHEMA Procedure (page 2-17).
`Name`	Application name of the imported application. See GET_APPLICATION_NAME Function (page 2-7), SET_APPLICATION_NAME Procedure (page 2-13).
`Alias`	Application alias of the imported application. See GET_APPLICATION_ALIAS Function (page 2-6), SET_APPLICATION_ALIAS Procedure (page 2-11).
`Image Prefix`	The image prefix of the imported application. See GET_IMAGE_PREFIX Function (page 2-8), SET_IMAGE_PREFIX Procedure (page 2-14).
`Proxy`	The proxy server attributes of the imported application. See GET_PROXY Function (page 2-10), SET_PROXY Procedure (page 2-16).

2.3 Import Script Examples

Using the workspace FRED_DEV on the development instance, you generate an application export of application 645 and save it as file f645.sql. All examples in this section assume you are connected to SQL*Plus.

Import Application without Modification

To import this application back into the FRED_DEV workspace on the same development instance using the same application ID:

@f645.sql

Import Application with Specified Application ID

To import this application back into the FRED_DEV workspace on the same development instance, but using application ID 702:

```
begin
  apex_application_install.set_application_id( 702);
  apex_application_install.generate_offset;
  apex_application_install.set_application_alias( 'F' ||
apex_application.get_application_id );
end;
/
```

```
@645.sql
```

Import Application with Generated Application ID

To import this application back into the FRED_DEV workspace on the same development instance, but using an available application ID generated by Application Express:

```
begin
  apex_application_install.generate_application_id;
  apex_application_install.generate_offset;
  apex_application_install.set_application_alias( 'F' ||
apex_application.get_application_id );
end;
/

@f645.sql
```

Import Application into Different Workspace using Different Schema

To import this application into the FRED_PROD workspace on the production instance, using schema FREDDY, and the workspace ID of FRED_DEV and FRED_PROD are different:

```
begin
    apex_application_install.set_workspace('FRED_PROD');
    apex_application_install.generate_offset;
    apex_application_install.set_schema( 'FREDDY' );
    apex_application_install.set_application_alias( 'FREDPROD_APP' );
end;
/

@f645.sql
```

Import into Training Instance for Three Different Workspaces

To import this application into the Training instance for 3 different workspaces:

```
begin
    apex_application_install.set_workspace('TRAINING1');
    apex_application_install.generate_application_id;
    apex_application_install.generate_offset;
    apex_application_install.set_schema( 'STUDENT1' );
    apex_application_install.set_application_alias( 'F' ||
apex_application.get_application_id );
end;
/

@f645.sql

begin
    apex_application_install.set_workspace('TRAINING2');
    apex_application_install.generate_application_id;
    apex_application_install.generate_offset;
    apex_application_install.set_schema( 'STUDENT2' );
    apex_application_install.set_application_alias( 'F' ||
apex_application.get_application_id );
end;
/
```

```
@f645.sql

begin
    apex_application_install.set_workspace('TRAINING3');
    apex_application_install.generate_application_id;
    apex_application_install.generate_offset;
    apex_application_install.set_schema( 'STUDENT3' );
    apex_application_install.set_application_alias( 'F' ||
apex_application.get_application_id );
    end;
/

@f645.sql
```

2.4 CLEAR_ALL Procedure

This procedure clears all values currently maintained in the APEX_APPLICATION_INSTALL package.

Syntax

```
APEX_APPLICATION_INSTALL.CLEAR_ALL;
```

Parameters

None.

Example

The following example clears all values currently set by the APEX_APPLICATION_INSTALL package.

```
begin
    apex_application_install.clear_all;
end;
```

2.5 GENERATE_APPLICATION_ID Procedure

This procedure generates an available application ID on the instance and sets the application ID in APEX_APPLICATION_INSTALL.

Syntax

```
APEX_APPLICATION_INSTALL.GENERATE_APPLICATION_ID;
```

Parameters

None.

Example

For an example of this procedure call, see "Import Application with Generated Application ID (page 2-4)" and "Import into Training Instance for Three Different Workspaces (page 2-4)".

See Also:

- "SET_APPLICATION_ID Procedure (page 2-12)"
- "GET_APPLICATION_ID Function (page 2-7)"

2.6 GENERATE_OFFSET Procedure

This procedure generates the offset value used during application import. Use the offset value to ensure that the metadata for the Application Express application definition does not collide with other metadata on the instance. For a new application installation, it is usually sufficient to call this procedure to have Application Express generate this offset value for you.

Syntax

```
APEX_APPLICATION_INSTALL.GENERATE_OFFSET;
```

Parameters

None.

Example

For examples of this procedure call, see "Import Application with Specified Application ID (page 2-3)", "Import Application with Generated Application ID (page 2-4)", and "Import into Training Instance for Three Different Workspaces (page 2-4)".

See Also:

- "GET_OFFSET Function (page 2-9)"
- "SET_OFFSET Procedure (page 2-15)"

2.7 GET_APPLICATION_ALIAS Function

This function gets the application alias for the application to be imported. This is only used if the application to be imported has an alias specified. An application alias must be unique within a workspace and it is recommended to be unique within an instance.

Syntax

```
APEX_APPLICATION_INSTALL.GET_APPLICATION_ALIAS
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the value of the application alias value in the APEX_APPLICATION_INSTALL package. The application alias cannot be more than 255 characters.

```
declare
    l_alias varchar2(255);
begin
    l_alias := apex_application_install.get_application_alias;
end;
```

See Also:

"SET_APPLICATION_ALIAS Procedure (page 2-11)"

2.8 GET_APPLICATION_ID Function

Use this function to get the application ID of the application to be imported. The application ID should either not exist in the instance or, if it does exist, must be in the workspace where the application is being imported to.

Syntax

```
APEX_APPLICATION_INSTALL.GET_APPLICATION_ID
RETURN NUMBER;
```

Parameters

None.

Example

The following example returns the value of the application ID value in the APEX_APPLICATION_INSTALL package.

```
declare
    l_id number;
begin
    l_id := apex_application_install.get_application_id;
end;
```

See Also:

- "SET_APPLICATION_ID Procedure (page 2-12)"
- "GENERATE_APPLICATION_ID Procedure (page 2-5)"

2.9 GET_APPLICATION_NAME Function

This function gets the application name of the import application.

Syntax

```
APEX_APPLICATION_INSTALL.GET_APPLICATION_NAME
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the value of the application name value in the APEX_APPLICATION_INSTALL package.

```
declare
    l_application_name varchar2(255);
begin
    l_application_name := apex_application_install.get_application_name;
end;
```

See Also:

"SET_APPLICATION_NAME Procedure (page 2-13)"

2.10 GET_AUTO_INSTALL_SUP_OBJ Function

Use this function to get the automatic install of supporting objects setting used during the import of an application. This setting is valid only for command line installs. If the setting is set to TRUE and the application export contains supporting objects, it automatically installs or upgrades the supporting objects when an application imports from the command line.

Syntax

```
APEX_APPLICATION_INSTALL.GET_AUTO_INSTALL_SUP_OBJ
RETURN BOOLEAN;
```

Parameters

None.

Example

The following example returns the value of automatic install of supporting objects setting in the APEX_APPLICATION_INSTALL package.

```
declare
    l_auto_install_sup_obj boolean;
begin
    l_auto_install_sup_obj := apex_application_install.get_auto_install_sup_obj;
end;
```

2.11 GET_IMAGE_PREFIX Function

This function gets the image prefix of the import application. Most Application Express instances use the default image prefix of /i/.

Syntax

```
APEX_APPLICATION_INSTALL.GET_IMAGE_PREFIX
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the value of the application image prefix in the APEX_APPLICATION_INSTALL package. The application image prefix cannot be more than 255 characters.

```
declare
    l_image_prefix varchar2(255);
begin
    l_image_prefix := apex_application_install.get_image_prefix;
end;
```

See Also:

"SET_IMAGE_PREFIX Procedure (page 2-14)"

2.12 GET_KEEP_SESSIONS Function

This function finds out if sessions and session state will be preserved or deleted on upgrades.

Syntax

```
function get_keep_sessions
    return boolean
```

Example

The following example shows whether print sessions will be kept or deleted.

```
dbms_output.put_line (
    case when apex_application_install.get_keep_sessions then 'sessions will be kept'
    else 'sessions will be deleted'
    end );
```

See Also:

"SET_KEEP_SESSIONS Procedure (page 2-15)"

2.13 GET_OFFSET Function

Use function to get the offset value used during the import of an application.

Syntax

```
APEX_APPLICATION_INSTALL.GET_OFFSET
RETURN NUMBER;
```

Parameters

None.

Example

The following example returns the value of the application offset value in the APEX_APPLICATION_INSTALL package.

```
declare
    l_offset number;
begin
    l_offset := apex_application_install.get_offset;
end;
```

See Also:

- "SET_OFFSET Procedure (page 2-15)"
- "GENERATE_OFFSET Procedure (page 2-6)"

2.14 GET_PROXY Function

Use this function to get the proxy server attribute of an application to be imported.

Syntax

```
APEX_APPLICATION_INSTALL.GET_PROXY
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the value of the proxy server attribute in the APEX_APPLICATION_INSTALL package. The proxy server attribute cannot be more than 255 characters.

```
declare
    l_proxy varchar2(255);
begin
    l_proxy := apex_application_install.get_proxy;
end;
```

See Also:

"SET_PROXY Procedure (page 2-16)"

2.15 GET_SCHEMA Function

Use this function to get the parsing schema ("owner") of the Application Express application.

Syntax

```
APEX_APPLICATION_INSTALL.GET_SCHEMA
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the value of the application schema in the APEX_APPLICATION_INSTALL package.

```
declare
    l_schema varchar2(30);
begin
    l_schema := apex_application_install.get_schema;
end;
```

See Also:

"SET_SCHEMA Procedure (page 2-17)"

2.16 GET_WORKSPACE_ID Function

Use this function to get the workspace ID for the application to be imported.

Syntax

```
APEX_APPLICATION_INSTALL.GET_WORKSPACE_ID
RETURN NUMBER;
```

Parameters

None.

Example

The following example returns the value of the workspace ID value in the APEX_APPLICATION_INSTALL package.

```
declare
    l_workspace_id number;
begin
    l_workspace_id := apex_application_install.get_workspace_id;
end;
```

See Also:

"SET_WORKSPACE_ID Procedure (page 2-17)"

2.17 SET_APPLICATION_ALIAS Procedure

This procedure sets the application alias for the application to be imported. This is only used if the application to be imported has an alias specified. An application alias must be unique within a workspace and it is recommended to be unique within an instance.

Syntax

```
APEX_APPLICATION_INSTALL.SET_APPLICATION_ALIAS(
    p_application_alias IN VARCHAR2);
```

Parameters

Table 2-2 SET_APPLICATION_ALIAS Parameters

Parameter	Description
p_application_alias	The application alias. The application alias is an alphanumeric identifier. It cannot exceed 255 characters, must unique within a workspace and, ideally, is unique within an entire instance.

Example

For examples of this procedure call, see "Import Application with Specified Application ID (page 2-3)", "Import Application with Generated Application ID (page 2-4)", "Import Application into Different Workspace using Different Schema (page 2-4)" and "Import into Training Instance for Three Different Workspaces (page 2-4)".

See Also:

"GET_APPLICATION_ALIAS Function (page 2-6)"

2.18 SET_APPLICATION_ID Procedure

Use this procedure to set the application ID of the application to be imported. The application ID should either not exist in the instance or, if it does exist, must be in the workspace where the application is being imported to. This number must be a positive integer and must not be from the reserved range of Application Express application IDs.

Syntax

```
APEX_APPLICATION_INSTALL.SET_APPLICATION_ID (
    p_application_id IN NUMBER);
```

Parameters

Table 2-3 SET_APPLICATION_ID Parameters

Parameter	Description
p_application_id	This is the application ID. The application ID must be a positive integer, and cannot be in the reserved range of application IDs (3000 - 8999). It must be less than 3000 or greater than or equal to 9000.

Example

For an example of this procedure call, see "Import Application with Specified Application ID (page 2-3)".

See Also:

- "SET_APPLICATION_ID Procedure (page 2-12)"
- "GENERATE_APPLICATION_ID Procedure (page 2-5)"

2.19 SET_APPLICATION_NAME Procedure

This procedure sets the application name of the import application.

Syntax

```
APEX_APPLICATION_INSTALL.SET_APPLICATION_NAME;(
    p_application_name  IN VARCHAR2);
```

Parameters

Table 2-4 SET_APPLICATION_NAME Parameters

Parameter	Description
p_application_name	This is the application name. The application name cannot be null and cannot be longer than 255 characters.

Example

The following example sets the application name in APEX_APPLICATION_INSTALL to "Executive Dashboard".

```
declare
    l_name varchar2(255) := 'Executive Dashboard';
begin
    apex_application_install.set_application_name( p_application_name => l_name );
end;
```

See Also:

"GET_APPLICATION_NAME Function (page 2-7)"

2.20 SET_AUTO_INSTALL_SUP_OBJ Procedure

This procedure sets the automatic install of supporting objects value used during application import. This setting is valid only for command line installs. If the value is set to TRUE and the application export contains supporting objects, it automatically installs or upgrades the supporting objects when an application imports from the command line.

Syntax

```
APEX_APPLICATION_INSTALL.SET_AUTO_INSTALL_SUP_OBJ(
    p_auto_install_sup_obj IN BOOLEAN);
```

Parameters

Table 2-5 SET_AUTO_INSTALL_SUP_OBJ Parameters

Parameter	Description
p_auto_install_sup_obj	The automatic install of supporting objects Boolean value.

Example

The following example gets the automatic install of supporting objects setting. If it is not set to install automatically, it sets to `true` to override export file settings of automatic install of supporting objects.

```
begin
    apex_application_install.set_auto_install_sup_obj( p_auto_install_sup_obj =>
true );
end;
```

2.21 SET_IMAGE_PREFIX Procedure

This procedure sets the image prefix of the import application. Most Application Express instances use the default image prefix of `/i/`.

Syntax

```
APEX_APPLICATION_INSTALL.SET_IMAGE_PREFIX(
    p_image_prefix  IN VARCHAR2);
```

Parameters

Table 2-6 SET_AUTO_INSTALL_SUP_OBJ Parameters

Parameter	Description
p_auto_install_sup_obj	The automatic install of supporting objects Boolean value.

Example

The following example sets the value of the image prefix variable in `APEX_APPLICATION_INSTALL`.

```
declare
    l_prefix varchar2(255) := '/i/';
begin
    apex_application_install.set_image_prefix( p_image_prefix => l_prefix );
end;
```

See Also:

"GET_IMAGE_PREFIX Function (page 2-8)"

2.22 SET_KEEP_SESSIONS Procedure

This procedure preserves sessions associated with the application on upgrades.

Syntax

```
procedure set_keep_sessions (
    p_keep_sessions in boolean );
```

Parameters

Table 2-7 SET_KEEP_SESSIONS Parameters

Parameter	Description
p_keep_sessions	false is the default value.true if sessions should be preserved, false if they should be deleted. KEEP_SESSIONS_ON_UPGRADE controls the default behavior. If it is N (the default), sessions will be deleted. KEEP_SESSIONS_ON_UPGRADE is an instance parameter.

Example

The following example installs application 100 in workspace FRED_PROD and keep session state.

```
SQL> exec apex_application_install.set_workspace(p_workspace => 'FRED_PROD');
SQL> exec apex_application_install.set_keep_sessions(p_keep_sessions => true);
SQL> @f100.sql
```

See Also:

"GET_KEEP_SESSIONS Function (page 2-9)"

2.23 SET_OFFSET Procedure

This procedure sets the offset value used during application import. Use the offset value to ensure that the metadata for the Application Express application definition does not collide with other metadata on the instance. For a new application installation, it is usually sufficient to call the generate_offset procedure to have Application Express generate this offset value for you.

Syntax

```
APEX_APPLICATION_INSTALL.SET_OFFSET(
    p_offset IN NUMBER);
```

Parameters

Table 2-8 SET_OFFSET Parameters

Parameter	Description
p_offset	The offset value. The offset must be a positive integer. In most cases you do not need to specify the offset, and instead, call APEX_APPLICATION_INSTALL.GENERATE_OFFSET, which generates a large random value and then set it in the APEX_APPLICATION_INSTALL package.

Example

The following example generates a random number from the database and uses this as the offset value in APEX_APPLICATION_INSTALL.

```
declare
    l_offset number;
begin
    l_offset := dbms_random.value(100000000000, 999999999999);
    apex_application_install.set_offset( p_offset => l_offset );
end/
```

See Also:

- "GET_OFFSET Function (page 2-9)"
- "GENERATE_OFFSET Procedure (page 2-6)"

2.24 SET_PROXY Procedure

Use this procedure to set the proxy server attributes of an application to be imported.

Syntax

```
APEX_APPLICATION_INSTALL.SET_PROXY (
    p_proxy IN VARCHAR2);
```

Parameters

Table 2-9 SET_PROXY Parameters

Parameter	Description
p_proxy	The proxy server. There is no default value. The proxy server cannot be more than 255 characters and should not include any protocol prefix such as http://. A sample value might be: www-proxy.company.com

Example

The following example sets the value of the proxy variable in APEX_APPLICATION_INSTALL.

```
declare
    l_proxy varchar2(255) := 'www-proxy.company.com'
begin
    apex_application_install.set_proxy( p_proxy => l_proxy );
end;
```

See Also:

"SET_PROXY Procedure (page 2-16)"

2.25 SET_SCHEMA Procedure

Use this function to set the parsing schema ("owner") of the Application Express application. The database user of this schema must already exist, and this schema name must already be mapped to the workspace used to import the application.

Syntax

```
APEX_APPLICATION_INSTALL.SET_SCHEMA (
    p_schema  IN VARCHAR2);
```

Parameters

Table 2-10 SET_SCHEMA Parameters

Parameter	Description
p_schema	The schema name.

Example

For examples of this procedure call, see "Import Application into Different Workspace using Different Schema (page 2-4)" and "Import into Training Instance for Three Different Workspaces (page 2-4)".

See Also:

"GET_SCHEMA Function (page 2-10)"

2.26 SET_WORKSPACE_ID Procedure

Use this function to set the workspace ID for the application to be imported.

Syntax

```
APEX_APPLICATION_INSTALL.SET_WORKSPACE_ID (
    p_workspace_id  IN NUMBER);
```

Parameters

Table 2-11 SET_WORKSPACE_ID Parameters

Parameter	Description
p_workspace_id	The workspace ID.

Example

For examples of this procedure call, see "Import Application into Different Workspace using Different Schema (page 2-4)" and "Import into Training Instance for Three Different Workspaces (page 2-4)".

See Also:

"SET_WORKSPACE_ID Procedure (page 2-17)"

2.27 SET_WORKSPACE_Procedure

This function is used to set the workspace ID for the application to be imported.

Syntax

```
procedure set_workspace (
   p_workspace in varchar2 );
```

Parameters

Table 2-12 SET_WORKSPACE_Procedure Parameters

Parameters	Description
p_workspace	The workspace name.

Example

This example shows how to set workspace ID for workspace FRED_PROD.

```
apex_application_install.set_workspace (
    p_workspace => 'FRED_PROD' );
```

See Also:

- "GET_WORKSPACE_ID Function (page 2-11)"
- "SET_WORKSPACE_ID Procedure (page 2-17)"

3

APEX_AUTHENTICATION

The APEX_AUTHENTICATION package provides a public API for authentication plugins.

3.1 Constants

The following constant is used by this package.

```
c_default_username_cookie constant varchar2(30) := 'LOGIN_USERNAME_COOKIE';
```

3.2 CALLBACK Procedure

This procedure is the landing resource for external login pages. Call this procedure directly from the browser.

Syntax

```
APEX_AUTHENTICATION.CALLBACK (
    p_session_id IN NUMBER,
    p_app_id IN NUMBER,
    p_page_id IN NUMBER DEFAULT NULL,
    p_ajax_identifier IN VARCHAR2,
    p_x01 IN VARCHAR2 DEFAULT NULL,
    p_x02 IN VARCHAR2 DEFAULT NULL,
    p_x03 IN VARCHAR2 DEFAULT NULL,
    p_x04 IN VARCHAR2 DEFAULT NULL,
    p_x05 IN VARCHAR2 DEFAULT NULL,
    p_x06 IN VARCHAR2 DEFAULT NULL,
    p_x07 IN VARCHAR2 DEFAULT NULL,
    p_x08 IN VARCHAR2 DEFAULT NULL,
```

```
    p_x09 IN VARCHAR2 DEFAULT NULL,
    p_x10 IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 3-1 APEX_AUTHENTICATION.CALLBACK Procedure Parameters

Parameters	Description
`p_session_id`	The Application Express session identifier.
`p_app_id`	The database application identifier.
`p_page_id`	Optional page identifier.
`p_ajax_identifierp`	The system generated Ajax identifier. See "GET_AJAX_IDENTIFIER Function (page 20-9)."
`p_x01` through `p_x10`	Optional parameters that the external login passes to the authentication plugin.

Example 1

In this example, a redirect is performed to an external login page and the callback is passed into Application Express, which the external login redirects to after successful authentication.

```
declare
    l_callback varchar2(4000) := apex_application.get_callback_url;
begin
    sys.owa_util.redirect_url(
        'https://single-signon.example.com/my_custom_sso.login?p_on_success='||
        sys.utl_url.escape (
            url => l_callback,
            escape_reserved_chars => true );
    apex_application.stop_apex_engine;
end;
```

Example 2

In this example, an external login page saves user data in a shared table and performs a call back with a handle to the data. In Application Express, the callback activates the authentication plugin's ajax code. It can take the value of `x01` and fetch the actual user data from the shared table.

```
---- create or replace package body my_custom_sso as
procedure login (
    p_on_success in varchar2 )
    is
    l_login_id varchar2(32);
begin
    l_login_id := rawtohex(sys.dbms_crypto.random(32));
    insert into login_data(id, username) values (l_login_id, 'JOE USER');
    sys.owa_util.redirect_url (
    p_on_success||'&p_x01='||l_login_id );
end;
---- end my_custom_sso;
```

Note:

"GET_CALLBACK_URL Function (page 3-3)"

3.3 GET_CALLBACK_URL Function

This function is a plugin helper function to return a URL that is used as a landing request for external login pages. When the browser sends the request, it triggers the authentication plugin ajax callback, which can be used to log the user in.

Syntax

```
APEX_AUTHENTICATION.GET_CALLBACK_URL (
    p_x01 IN VARCHAR2 DEFAULT NULL,
    p_x02 IN VARCHAR2 DEFAULT NULL,
    p_x03 IN VARCHAR2 DEFAULT NULL,
    p_x04 IN VARCHAR2 DEFAULT NULL,
    p_x05 IN VARCHAR2 DEFAULT NULL,
    p_x06 IN VARCHAR2 DEFAULT NULL,
    p_x07 IN VARCHAR2 DEFAULT NULL,
    p_x08 IN VARCHAR2 DEFAULT NULL,
    p_x09 IN VARCHAR2 DEFAULT NULL,
    p_x10 IN VARCHAR2 DEFAULT NULL )
    return VARCHAR2;
```

Parameters

Table 3-2 APEX_AUTHENTICATION.GET_CALLBACK _URL Function Parameters

Parameters	Description
`p_x01` through `p_x10`	Optional parameters that the external login passes to the authentication plugin.

Example

See Also:

"CALLBACK Procedure (page 3-1)"

3.4 GET_LOGIN_USERNAME_COOKIE Function

This function reads the cookie with the username from the default login page.

Syntax

```
APEX_AUTHENTICATION.GET_LOGIN_USERNAME_COOKIE (
    p_cookie_name IN VARCHAR2 DEFAULT c_default_username_cookie )
    return varchar2;
```

Parameters

Table 3-3 APEX_AUTHENTICATION.GET_LOGIN_USERNAME_COOKIE Function Parameters

Parameters	Description
p_cookie_name	The cookie name that stores the username in the browser.

Example

In this example, GET_LOGIN_USERNAME_COOKIE saves the username cookie value into the page item P101_USERNAME.

```
:P101_USERNAME := apex_authentication.get_login_username_cookie;
```

3.5 IS_AUTHENTICATED Function

This function checks if the user is authenticated in the session and returns TRUE if the user is already logged in or FALSE if the user of the current session is not yet authenticated.

Syntax

```
APEX_AUTHENTICATION.IS_AUTHENTICATED
    RETURN BOOLEAN;
```

Parameters

None.

Example

In this example, IS_AUTHENTICATED is used to emit the username if the user has already logged in or a notification if the user has not.

```
if apex_authentication.is_authenticated then
    sys.htp.p(apex_escape.html(:APP_USER)||', you are known to the system');
else
    sys.htp.p('Please sign in');
end if;
```

Note:

"IS_PUBLIC_USER Function (page 3-4)"

3.6 IS_PUBLIC_USER Function

This function checks if the user is not authenticated in the session. A FALSE is returned if the user is already logged on or TRUE if the user of the current session is not yet authenticated.

Syntax

```
APEX_AUTHENTICATION.IS_PUBLIC_USER
    return BOLLEAN;
```

Parameters

None.

Example

In this example, `IS_PUBLIC_USER` is used to show a notification if the user has not already logged in or the username if the user has not.

```
if apex_authentication.is_public_user then
    sys.htp.p('Please sign in');
else
    sys.htp.p(apex_escape.html(:APP_USER)||', you are known to the system');
end if;
```

3.7 LOGIN Procedure

This procedure authenticates the user in the current session.

Login processing has the following steps:

1. Run authentication scheme's pre-authentication procedure.
2. Run authentication scheme's authentication function to check the user credentials (p_username, p_password), returning TRUE on success.
3. If result=true: run post-authentication procedure.
4. If result=true: save username in session table.
5. If result=true: set redirect url to deep link.
6. If result=false: set redirect url to current page, with an error message in the notification_msg parameter.
7. Log authentication result.
8. Redirect.

Syntax

```
APEX_AUTHENTICATION.LOGIN (
    p_username IN VARCHAR2,
    p_password IN VARCHAR2,
    p_uppercase_username IN BOOLEAN DEFAULT TRUE );
```

Parameters

Table 3-4 APEX_AUTHENTICATION.LOGIN Procedure Parameters

Parameters	Description
`p_username`	The user's name.
`p_password`	The user's password.
`p_uppercase_username`	If `TRUE` then `p_username` is converted to uppercase.

Example

This example passes user credentials, username and password, to the authentication scheme.

```
apex_authentication.login('JOE USER', 'mysecret');
```

Note:

"POST_LOGIN Procedure (page 3-6)"

3.8 LOGOUT Procedure

This procedure closes the session and redirects to the application's home page. Call this procedure directly from the browser.

Syntax

```
APEX_AUTHENTICATION.LOGOUT (
    p_session_id in number,
    p_app_id in number,
    p_ws_app_id in number default null );
```

Parameters

Table 3-5 APEX_AUTHENTICATION.LOGOUT Procedure Parameters

Parameters	Description
p_session_id	The Application Express session identifier of the session to close.
p_app_id	The database application identifier.
p_ws_app_id	The websheet application identifier.

Example

This example logs the session out.

```
apex_authentication.logout(:SESSION, :APP_ID);
```

3.9 POST_LOGIN Procedure

This procedure authenticates the user in the current session. It runs a subset of login(), without steps 1 and 2. For steps, see "LOGIN Procedure (page 3-5)." It is primarily useful in authentication schemes where user credentials checking is done externally to Application Express.

Syntax

```
APEX_AUTHENTICATION.POST_LOGIN (
    p_username IN VARCHAR2,
    p_password IN VARCHAR2,
    p_uppercase_username IN BOOLEAN DEFAULT TRUE );
```

Parameters

Table 3-6 APEX_AUTHENTICATION.POST_LOGIN Procedure Parameters

Parameters	Description
p_username	The user's name.
p_password	The user's password.
p_uppercase_username	If TRUE then p_username is converted to uppercase.

Example

This procedure call passes user credentials, username and password, to the authentication scheme to finalize the user's authentication.

```
apex_authentication.post_login('JOE USER', 'mysecret');
```

Note:

"LOGIN Procedure (page 3-5)"

3.10 SEND_LOGIN_USERNAME_COOKIE Procedure

This procedure sends a cookie with the username.

Syntax

```
APEX_AUTHENTICATION.SEND_LOGIN_USERNAME_COOKIE (
    p_username IN VARCHAR2,
    p_cookie_name IN VARCHAR2 DEFAULT c_default_username_cookie );
```

Parameters

Table 3-7 APEX_AUTHENTICATION.SEND_LOGIN_USERNAME_COOKIE Procedure Parameters

Parameters	Description
p_username	The user's name.
p_cookie_name	The cookie name which stores p_username in the browser.

Example

This example shows how to call SEND_LOGIN_USERNAME_COOKIE, to save the value of item P101_USERNAME in the login cookie.

```
apex_authentication.send_login_username_cookie (
    p_username => :P101_USERNAME );
```

4

APEX_AUTHORIZATION

The APEX_AUTHORIZATION package contains public utility functions used for controlling and querying access rights to the application.

4.1 ENABLE_DYNAMIC_GROUPS Procedure

This procedure enables groups in the current session. These groups do not have to be created in the Application Express workspace repository, but can, for example, be loaded from a LDAP repository. Enabling a group that exists in the workspace repository and has other groups granted to it, also enables the granted groups.

If Real Application Security, available with Oracle Database Release 12*g*, is enabled for the authentication scheme, all dynamic groups are enabled as RAS dynamic or external groups (depending whether the group exists in dba_xs_dynamic_roles).

This procedure must be called during or right after authentication, for example, in a post-authentication procedure.

Syntax

```
APEX_AUTHORIZATION.ENABLE_DYNAMIC_GROUPS (
    p_group_names IN apex_t_varchar2 );
```

Parameters

Table 4-1 ENABLE_DYNAMIC_GROUPS Procedure Parameter

Parameter	Description
p_group_names	Table of group names.

Example

This example enables the dynamic groups SALES and HR, for example, from within a post authentication procedure.

```
begin
   apex_authorization.enable_dynamic_groups (
       p_group_names => apex_t_varchar2('SALES', 'HR') );
end;
```

See Also:

View APEX_WORKSPACE_SESSION_GROUPS and View APEX_WORKSPACE_GROUP_GROUPS

4.2 IS_AUTHORIZED Function

Determine if the current user passes the authorization with name p_authorization_name. For performance reasons, authorization results are cached. Because of this, the function may not always evaluate the authorization when called, but take the result out of the cache.

See Also:

"Changing the Evaluation Point Attribute" in *Oracle Application Express App Builder User's Guide*

Syntax

```
APEX_AUTHORIZATION.IS_AUTHORIZED (
    p_authorization_name IN VARCHAR2 )
    RETURN BOOLEAN;
```

Parameters

Table 4-2 IS_AUTHORIZED Function Parameters

Parameter	Description
p_authorization_name	The name of an authorization scheme in the application.

Returns

Table 4-3 IS_AUTHORIZED Function Returns

Parameter	Description
TRUE	If the authorization is successful.
FALSE	If the authorization is not successful.

Example

This example prints the result of the authorization "User Is Admin".

```
begin
    sys.htp.p('User Is Admin: '||
              case apex_authorization.is_authorized (
                       p_authorization_name => 'User Is Admin' )
              when true then 'YES'
              when false then 'NO'
              else 'null'
              end);
end;
```

4.3 RESET_CACHE Procedure

This procedure resets the authorization caches for the session and forces a re-evaluation when an authorization is checked next.

Syntax

```
APEX_AUTHORIZATION.RESET_CACHE;
```

Parameters

None.

Example

This examples resets the authorization cache.

```
apex_authorization.reset_cache;
```

5

APEX_COLLECTION

Collections enable you to temporarily capture one or more nonscalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed during a user's specific session. You can think of a collection as a bucket in which you temporarily store and name rows of information.

5.1 About the APEX_COLLECTION API

Every collection contains a named list of data elements (or members) which can have up to 50 character attributes (`VARCHAR2(4000)`), five number attributes, five date attributes, one XML Type attribute, one large binary attribute (`BLOB`), and one large character attribute (`CLOB`). You insert, update, and delete collection information using the PL/SQL API `APEX_COLLECTION`.

The following are examples of when you might use collections:

- When you are creating a data-entry wizard in which multiple rows of information first need to be collected within a logical transaction. You can use collections to temporarily store the contents of the multiple rows of information, before performing the final step in the wizard when both the physical and logical transactions are completed.
- When your application includes an update page on which a user updates multiple detail rows on one page. The user can make many updates, apply these updates to a collection and then call a final process to apply the changes to the database.
- When you are building a wizard where you are collecting an arbitrary number of attributes. At the end of the wizard, the user then performs a task that takes the information temporarily stored in the collection and applies it to the database.

Beginning in Oracle Database 12*c*, database columns of data type `VARCHAR2` can be defined up to 32,767 bytes. This requires that the database initialization parameter `MAX_STRING_SIZE` has a value of `EXTENDED`. If Application Express was installed in Oracle Database 12*c* and with `MAX_STRING_SIZE` = `EXTENDED`, then the tables for the Application Express collections will be defined to support up 32,767 bytes for the character attributes of a collection. For the methods in the APEX_COLLECTION API, all references to character attributes (`c001` through `c050`) can support up to 32,767 bytes.

5.2 Naming Collections

When you create a collection, you must give it a name that cannot exceed 255 characters. Note that collection names are not case-sensitive and are converted to uppercase.

Once the collection is named, you can access the values in the collection by running a SQL query against the view `APEX_COLLECTIONS`.

See Also:

- "Accessing a Collection (page 5-5)"
- "Table 5-7 (page 5-17)"
- "Table 5-8 (page 5-18)"

5.3 Creating a Collection

Every collection contains a named list of data elements (or members) which can have up to 50 character attributes (`VARCHAR2(4000)`), five number attributes, one XML Type attribute, one large binary attribute (`BLOB`), and one large character attribute (`CLOB`). You use the following methods to create a collection:

- `CREATE_COLLECTION`

  This method creates an empty collection with the provided name. An exception is raised if the named collection exists.

- `CREATE_OR_TRUNCATE_COLLECTION`

  If the provided named collection does not exist, this method creates an empty collection with the given name. If the named collection exists, this method truncates it. Truncating a collection empties it, but leaves it in place.

- `CREATE_COLLECTION_FROM_QUERY`

  This method creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. This method can be used with a query with up to 50 columns in the `SELECT` clause. These columns in the `SELECT` clause populate the 50 character attributes of the collection (C001 through C050).

- `CREATE_COLLECTION_FOM_QUERY2`

  This method creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. It is identical to the `CREATE_COLLECTION_FROM_QUERY`, however, the first 5 columns of the `SELECT` clause must be numeric. After the numeric columns, there can be up to 50 character columns in the `SELECT` clause.

- `CREATE_COLLECTION_FROM_QUERY_B`

  This method offers significantly faster performance than the `CREATE_COLLECTION_FROM_QUERY` method by performing bulk SQL operations, but has the following limitations:

  – No column value in the select list of the query can be more than 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error is raised during execution.

  – The MD5 checksum is not computed for any members in the collection.

- `CREATE_COLLECTION_FROM_QUERYB2`

  This method also creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. It is identical to the `CREATE_COLLECTION_FROM_QUERY_B`, however, the first five columns of the `SELECT` clause must be numeric. After the numeric columns, there can be up to 50 character columns in the `SELECT` clause.

See Also:

- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.4 About the Parameter p_generate_md5

Use the `p_generate_md5` flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to `NO`. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

See Also:

- "Determining Collection Status (page 5-10)"for information about using the `GET_MEMBER_MD5` function
- "GET_MEMBER_MD5 Function (page 5-30)"

5.5 Accessing a Collection

You can access the members of a collection by querying the database view `APEX_COLLECTIONS`. Collection names are always converted to uppercase. When querying the `APEX_COLLECTIONS` view, always specify the collection name in all uppercase. The `APEX_COLLECTIONS` view has the following definition:

```
COLLECTION_NAME   NOT NULL VARCHAR2(255)
SEQ_ID            NOT NULL NUMBER
C001              VARCHAR2(4000)
C002              VARCHAR2(4000)
C003              VARCHAR2(4000)
C004              VARCHAR2(4000)
C005              VARCHAR2(4000)
...
C050              VARCHAR2(4000)
N001              NUMBER
N002              NUMBER
N003              NUMBER
N004              NUMBER
N005              NUMBER
CLOB001           CLOB
BLOB001           BLOB
XMLTYPE001        XMLTYPE
MD5_ORIGINAL      VARCHAR2(4000)
```

Use the `APEX_COLLECTIONS` view in an application just as you would use any other table or view in an application, for example:

```
SELECT c001, c002, c003, n001, clob001
   FROM APEX_collections
 WHERE collection_name = 'DEPARTMENTS'
```

5.6 Merging Collections

You can merge members of a collection with values passed in a set of arrays. By using the `p_init_query` argument, you can create a collection from the supplied query.

See Also:

"MERGE_MEMBERS Procedure (page 5-31)"

5.7 Truncating a Collection

If you truncate a collection, you remove all members from the specified collection, but the named collection remains in place.

See Also:

"TRUNCATE_COLLECTION Procedure (page 5-37)"

5.8 Deleting a Collection

If you delete a collection, you delete the collection and all of its members. Be aware that if you do not delete a collection, it is eventually deleted when the session is purged.

See Also:

"DELETE_COLLECTION Procedure (page 5-28)"

5.9 Deleting All Collections for the Current Application

Use the `DELETE_ALL_COLLECTIONS` method to delete all collections defined in the current application.

See Also:

"DELETE_ALL_COLLECTIONS Procedure (page 5-26)"

5.10 Deleting All Collections in the Current Session

Use the `DELETE_ALL_COLLECTIONS_SESSION` method to delete all collections defined in the current session.

See Also:

"DELETE_ALL_COLLECTIONS_SESSION Procedure (page 5-27)"

5.11 Adding Members to a Collection

When data elements (or members) are added to a collection, they are assigned a unique sequence ID. As you add members to a collection, the sequence ID is change in increments of 1, with the newest members having the largest ID.

You add new members to a collection using the `ADD_MEMBER function`. Calling this function returns the sequence ID of the newly added member.

You can also add new members (or an array of members) to a collection using the `ADD_MEMBERS` procedure. The number of members added is based on the number of elements in the first array.

See Also:

- "ADD_MEMBER Procedure (page 5-10)"
- "ADD_MEMBER Function (page 5-12)"
- "ADD_MEMBERS Procedure (page 5-13)"

5.12 About the Parameters p_generate_md5, p_clob001, p_blob001, and p_xmltype001

Use the `p_generate_md5` flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to `NO`. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

Use `p_clob001` for collection member attributes which exceed 4,000 characters. Use `p_blob001` for binary collection member attributes. Use `p_xmltype001` to store well-formed XML.

See Also:

"Determining Collection Status (page 5-10)" for information about using the function `GET_MEMBER_MD5`

5.13 Updating Collection Members

You can update collection members by calling the `UPDATE_MEMBER` procedure and referencing the desired collection member by its sequence ID. The `UPDATE_MEMBER` procedure replaces an entire collection member, not individual member attributes.

Use the `p_clob001` parameter for collection member attributes which exceed 4,000 characters.

To update a single attribute of a collection member, use the `UPDATE_MEMBER_ATTRIBUTE procedure`.

See Also:

- "UPDATE_MEMBER Procedure (page 5-38)"
- "UPDATE_MEMBERS Procedure (page 5-40)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 (page 5-43)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 (page 5-44)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 (page 5-45)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 (page 5-47)"

5.14 Deleting Collection Members

You can delete a collection member by calling the `DELETE_MEMBER` procedure and referencing the desired collection member by its sequence ID. Note that this procedure leaves a gap in the sequence IDs in the specified collection.

You can also delete all members from a collection by when an attribute matches a specific value. Note that the `DELETE_MEMBERS` procedure also leaves a gap in the sequence IDs in the specified collection. If the supplied attribute value is null, then all members of the named collection are deleted where the attribute (specified by `p_attr_number`) is null.

See Also:

- "DELETE_MEMBER Procedure (page 5-29)"
- "DELETE_MEMBERS Procedure (page 5-29)"

5.15 Obtaining a Member Count

Use `COLLECTION_MEMBER_COUNT` to return the total count of all members in a collection. Note that this count does not indicate the highest sequence in the collection.

See Also:

"COLLECTION_MEMBER_COUNT Function (page 5-16)"

5.16 Resequencing a Collection

Use `RESEQUENCE_COLLECTION` to resequence a collection to remove any gaps in sequence IDs while maintaining the same element order.

See Also:

"RESEQUENCE_COLLECTION Procedure (page 5-35)"

5.17 Verifying Whether a Collection Exists

Use `COLLECTION_EXISTS` to determine if a collection exists.

See Also:

"COLLECTION_EXISTS Function (page 5-15)"

5.18 Adjusting a Member Sequence ID

You can adjust the sequence ID of a specific member within a collection by moving the ID up or down. When you adjust a sequence ID, the specified ID is exchanged with another ID. For example, if you were to move the ID 2 up, 2 becomes 3, and 3 would become 2.

Use `MOVE_MEMBER_UP` to adjust a member sequence ID up by one. Alternately, use `MOVE_MEMBER_DOWN` to adjust a member sequence ID down by one.

See Also:

- "MOVE_MEMBER_DOWN Procedure (page 5-33)"
- "MOVE_MEMBER_UP Procedure (page 5-34)"

5.19 Sorting Collection Members

Use the `SORT_MEMBERS` method to reorder members of a collection by the column number. This method sorts the collection by a particular column number and also reassigns the sequence IDs for each member to remove gaps.

See Also:

"SORT_MEMBERS Procedure (page 5-37)"

5.20 Clearing Collection Session State

Clearing the session state of a collection removes the collection members. A shopping cart is a good example of when you might need to clear collection session state. When a user requests to empty the shopping cart and start again, you must clear the session state for a collection. You can remove session state of a collection by calling the `TRUNCATE_COLLECTION` method or by using `f?p` syntax.

Calling the `TRUNCATE_COLLECTION` method deletes the existing collection and then recreates it, for example:

```
APEX_COLLECTION.TRUNCATE_COLLECTION(
    p_collection_name => collection name);
```

You can also use the sixth `f?p` syntax argument to clear session state, for example:

```
f?p=App:Page:Session::NO:collection name
```

See Also:

"TRUNCATE_COLLECTION Procedure (page 5-37)"

5.21 Determining Collection Status

The `p_generate_md5` parameter determines if the MD5 message digests are computed for each member of a collection. The collection status flag is set to `FALSE` immediately after you create a collection. If any operations are performed on the collection (such as add, update, truncate, and so on), this flag is set to `TRUE`.

You can reset this flag manually by calling `RESET_COLLECTION_CHANGED`.

Once this flag has been reset, you can determine if a collection has changed by calling `COLLECTION_HAS_CHANGED`.

When you add a new member to a collection, an MD5 message digest is computed against all 50 attributes and the CLOB attribute if the `p_generated_md5` parameter is set to `YES`. You can access this value from the `MD5_ORIGINAL` column of the view `APEX_COLLECTION`. You can access the MD5 message digest for the current value of a specified collection member by using the function `GET_MEMBER_MD5`.

See Also:

- "RESET_COLLECTION_CHANGED Procedure (page 5-36)"
- "COLLECTION_HAS_CHANGED Function (page 5-15)"
- "GET_MEMBER_MD5 Function (page 5-30)"

5.22 ADD_MEMBER Procedure

Use this procedure to add a new member to an existing collection. An error is raised if the specified collection does not exist for the current user in the same session for the current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) adds the new member with a sequence ID of 9.

Syntax

```
APEX_COLLECTION.ADD_MEMBER (
    p_collection_name IN VARCHAR2,
    p_c001 IN VARCHAR2 default null,
    ...
    p_c050 IN VARCHAR2 default null,
    p_n001 IN NUMBER default null,
    p_n002 IN NUMBER default null,
    p_n003 IN NUMBER default null,
    p_n004 IN NUMBER default null,
    p_n005 IN NUMBER default null,
    p_d001 IN DATE default null,
    p_d002 IN DATE default null,
    p_d003 IN DATE default null,
    p_d004 IN DATE default null,
    p_d005 IN DATE default null,
    p_clob001 IN CLOB default empty_clob(),
```

```
    p_blob001 IN BLOB default empty_blob(),
    p_xmltype001 IN XMLTYPE default null,
    p_generate_md5 IN VARCHAR2  default 'NO');
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.

Table 5-1 ADD_MEMBER Procedure Parameters

Parameter	Description
`p_collection_name`	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
`p_n001 through p_n005`	Attribute value of the numeric attributes to be added.
`p_d001` through `p_d005`	Attribute value of the date attribute.
`p_clob001`	Use p_clob001 for collection member attributes that exceed 4,000 characters.
`p_blob001`	Use p_blob001 for binary collection member attributes.
`p_xmltype001`	Use p_xmltype001 to store well-formed XML.
`p_generate_md5`	Valid values include `YES` and `NO`. `YES` to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following is an example of the ADD_MEMBER procedure.

```
APEX_COLLECTION.ADD_MEMBER(
       p_collection_name => 'GROCERIES'
       p_c001            => 'Grapes',
       p_c002            => 'Imported',
       p_n001            => 125,
       p_d001            => sysdate );
END;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "ADD_MEMBER Function (page 5-12)"
- "ADD_MEMBERS Procedure (page 5-13)"

5.23 ADD_MEMBER Function

Use this function to add a new member to an existing collection. Calling this function returns the sequence ID of the newly added member. An error is raised if the specified collection does not exist for the current user in the same session for the current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) adds the new member with a sequence ID of 9.

Syntax

```
APEX_COLLECTION.ADD_MEMBER (
    p_collection_name IN VARCHAR2,
    p_c001 IN VARCHAR2 default null,
    ...
    p_c050 IN VARCHAR2 default null,
    p_n001 IN NUMBER default null,
    p_n002 IN NUMBER default null,
    p_n003 IN NUMBER default null,
    p_n004 IN NUMBER default null,
    p_n005 IN NUMBER default null,
    p_d001 IN DATE default null,
    p_d002 IN DATE default null,
    p_d003 IN DATE default null,
    p_d004 IN DATE default null,
    p_d005 IN DATE default null,
    p_clob001 IN CLOB default empty_clob(),
    p_blob001 IN BLOB default empty_blob(),
    p_xmltype001 IN XMLTYPE default null,
    p_generate_md5 IN VARCHAR2 default 'NO')
RETURN NUMBER;
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.

Table 5-2 ADD_MEMBER Function Parameters

Parameter	Description
p_collection_name	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.

Table 5-2 (Cont.) ADD_MEMBER Function Parameters

Parameter	Description
p_c001 through p_c050	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
p_n001 through p_n005	Attribute value of the numeric attributes to be added.
p_d001 through p_d005	Attribute value of the date attribute to be added.
p_clob001	Use p_clob001 for collection member attributes that exceed 4,000 characters.
p_blob001	Use p_blob001 for binary collection member attributes.
p_xmltype001	Use p_xmltype001 to store well-formed XML.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

```
DECLARE
    l_seq number;
BEGIN
    l_seq := APEX_COLLECTION.ADD_MEMBER(
                p_collection_name => 'GROCERIES'
                p_c001            => 'Grapes',
                p_c002            => 'Imported',
                p_n001            => 125,
                p_d001            => sysdate );
END;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "ADD_MEMBER Procedure (page 5-10)"
- "ADD_MEMBERS Procedure (page 5-13)"

5.24 ADD_MEMBERS Procedure

Use this procedure to add an array of members to a collection. An error is raised if the specified collection does not exist for the current user in the same session for the current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) adds the new member with a sequence ID of 9. The count of elements in the p_c001 PL/SQL table is used as the total number of items across all PL/SQL tables. For example, if p_c001.count is 2 and

p_c002.count is 10, only 2 members are added. If p_c001 is null an application error is raised.

Syntax

```
APEX_COLLECTION.ADD_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_c001 IN APEX_APPLICATION_GLOBAL.VC_ARR2 default empty_vc_arr,
    p_c002 IN APEX_APPLICATION_GLOBAL.VC_ARR2 default empty_vc_arr,
    p_c003 IN APEX_APPLICATION_GLOBAL.VC_ARR2 default empty_vc_arr,
    ...
    p_c050 IN APEX_APPLICATION_GLOBAL.VC_ARR2 default empty_vc_arr,
    p_n001 IN APEX_APPLICATION_GLOBAL.N_ARR default empty_n_arr,
    p_n002 IN APEX_APPLICATION_GLOBAL.N_ARR default empty_n_arr,
    p_n003 IN APEX_APPLICATION_GLOBAL.N_ARR default empty_n_arr,
    p_n004 IN APEX_APPLICATION_GLOBAL.N_ARR default empty_n_arr,
    p_n005 IN APEX_APPLICATION_GLOBAL.N_ARR default empty_n_arr,
    p_d001 IN APEX_APPLICATION_GLOBAL.D_ARR default empty_d_arr,
    p_d002 IN APEX_APPLICATION_GLOBAL.D_ARR default empty_d_arr,
    p_d003 IN APEX_APPLICATION_GLOBAL.D_ARR default empty_d_arr,
    p_d004 IN APEX_APPLICATION_GLOBAL.D_ARR default empty_d_arr,
    p_d005 IN APEX_APPLICATION_GLOBAL.D_ARR default empty_d_arr,
    p_generate_md5 IN VARCHAR2 default 'NO');
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-3 ADD_MEMBERS Procedure Parameters

Parameter	Description
p_collection_name	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
p_c001 through p_c050	Array of character attribute values to be added.
p_n001 through p_n005	Array of numeric attribute values to be added.
p_d001 through p_d005	Array of date attribute values to be added.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following example shows how to add two new members to the EMPLOYEE table.

```
Begin
    APEX_COLLECTION.ADD_MEMBERS(
```

```
        p_collection_name => 'EMPLOYEE',
        p_c001 => l_arr1,
        p_c002 => 1_arr2);
End;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "ADD_MEMBER Procedure (page 5-10)"
- "ADD_MEMBER Function (page 5-12)"

5.25 COLLECTION_EXISTS Function

Use this function to determine if a collection exists. A TRUE is returned if the specified collection exists for the current user in the current session for the current Application ID, otherwise FALSE is returned.

Syntax

```
APEX_COLLECTION.COLLECTION_EXISTS (
    p_collection_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 5-4 COLLECTION_EXISTS Function Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. The collection name is not case sensitive and is converted to upper case.

Example

The following example shows how to use the COLLECTION_EXISTS function to determine if the collection named EMPLOYEES exists.

```
Begin
    l_exists := APEX_COLLECTION.COLLECTION_EXISTS (
        p_collection_name => 'EMPLOYEES');
End;
```

5.26 COLLECTION_HAS_CHANGED Function

Use this function to determine if a collection has changed since it was created or the collection changed flag was reset.

Syntax

```
APEX_COLLECTION.COLLECTION_HAS_CHANGED (
    p_collection_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 5-5 COLLECTION_HAS_CHANGED Function Parameters

Parameter	Description
p_collection_name	The name of the collection. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

The following example shows how to use the COLLECTION_HAS_CHANGED function to determine if the EMPLOYEES collection has changed since it was created or last reset.

```
Begin
    l_exists := APEX_COLLECTION.COLLECTION_HAS_CHANGED (
       p_collection_name => 'EMPLOYEES');
End;
```

5.27 COLLECTION_MEMBER_COUNT Function

Use this function to get the total number of members for the named collection. If gaps exist, the total member count returned is not equal to the highest sequence ID in the collection. If the named collection does not exist for the current user in the current session, an error is raised.

Syntax

```
APEX_COLLECTION.COLLECTION_MEMBER_COUNT (
    p_collection_name IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 5-6 COLLECTION_MEMBER_COUNT Function Parameters

Parameter	Description
p_collection_name	The name of the collection.

Example

This example shows how to use the COLLECTION_MEMBER_COUNT function to get the total number of members in the DEPARTMENTS collection.

```
Begin
    l_count := APEX_COLLECTION.COLLECTION_MEMBER_COUNT( p_collection_name =>
'DEPARTMENTS');
End;
```

5.28 CREATE_COLLECTION Procedure

Use this procedure to create an empty collection that does not already exist. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION(
    p_collection_name IN VARCHAR2);
```

Parameters

Table 5-7 CREATE_COLLECTION Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.

Example

This example shows how to use the CREATE_COLLECTION procedure to create an empty collection named EMPLOYEES.

```
Begin
    APEX_COLLECTION.CREATE_COLLECTION(
        p_collection_name => 'EMPLOYEES');
End;
```

See Also:

- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.29 CREATE_OR_TRUNCATE_COLLECTION Procedure

Use this procedure to create a collection. If a collection exists with the same name for the current user in the same session for the current Application ID, all members of the collection are removed. In other words, the named collection is truncated.

Syntax

```
APEX_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
    p_collection_name IN VARCHAR2);
```

Parameters

Table 5-8 CREATE_OR_TRUNCATE_COLLECTION Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. All members of the named collection are removed if the named collection exists for the current user in the current session.

Example

This example shows how to use the CREATE_OR_TRUNCATE_COLLECTION procedure to remove all members in an existing collection named EMPLOYEES.

```
Begin
    APEX_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
        p_collection_name => 'EMPLOYEES');
End;
```

See Also:

- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.30 CREATE_COLLECTION_FROM_QUERY Procedure

Use this procedure to create a collection from a supplied query. The query is parsed as the application owner. This method can be used with a query with up to 50 columns in the SELECT clause. These columns in the SELECT clause populates the 50 character attributes of the collection (C001 through C050). If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_generate_md5 IN VARCHAR2 default 'NO');
```

Parameters

Table 5-9 CREATE_COLLECTION_FROM_QUERY Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERY procedure to create a collection named AUTO and populate it with data from the AUTOS table. Because p_generate_md5 is 'YES', the MD5 checksum is computed to allow comparisons to determine change status.

```
Begin
    l_query := 'select make, model, year from AUTOS';
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY (
        p_collection_name => 'AUTO',
        p_query => l_query,
        p_generate_md5 => 'YES');
End;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.31 CREATE_COLLECTION_FROM_QUERY2 Procedure

Use this procedure to create a collection from a supplied query. This method is identical to CREATE_COLLECTION_FROM_QUERY, however, the first 5 columns of the SELECT clause must be numeric and the next 5 must be date. After the numeric and date columns, there can be up to 50 character columns in the SELECT clause. The query is parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY2 (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_generate_md5 IN VARCHAR2 default 'NO');
```

Parameters

Table 5-10 CREATE_COLLECTION_FROM_QUERY2 Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERY2 procedure to create a collection named EMPLOYEE and populate it with data from the EMP table. The first five columns (mgr, sal, comm, deptno, and null) are all numeric. Because p_generate_md5 is 'NO', the MD5 checksum is not computed.

```
begin;
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY2 (
        p_collection_name => 'EMPLOYEE',
        p_query => 'select empno, sal, comm, deptno, null, hiredate, null, null,
null, null, ename, job, mgr from emp',
        p_generate_md5 => 'NO');
end;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.32 CREATE_COLLECTION_FROM_QUERY_B Procedure

Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the CREATE_COLLECTION_FROM_QUERY method. The query is parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

This procedure uses bulk dynamic SQL to perform the fetch and insert operations into the named collection. Two limitations are imposed by this procedure:

1. The MD5 checksum for the member data is not computed.
2. No column value in query p_query can exceed 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error is raised during execution. In Oracle Database 11g Release 2 (11.2.0.1) or later, this column limit is 4,000 bytes.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_names IN apex_application_global.vc_arr2,
    p_values IN apex_application_global.vc_arr2,
    p_max_row_count IN NUMBER DEFAULT null);
```

Parameters

Table 5-11 CREATE_COLLECTION_FROM_QUERY_B Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_names	Array of bind variable names used in the query statement.
p_values	Array of bind variable values used in the bind variables in the query statement.
p_max_row_count	Maximum number of rows returned from the query in p_query which should be added to the collection.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERY_B procedure to create a collection named EMPLOYEES and populate it with data from the emp table.

```
declare
     l_query varchar2(4000);
 Begin
     l_query := 'select empno, ename, job, sal from emp where deptno = :b1';
```

```
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B (
        p_collection_name => 'EMPLOYEES',
        p_query => l_query,
        p_names => apex_util.string_to_table('b1'),
        p_values => apex_util.string_to_table('10'));
End;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.33 CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version)

Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the `CREATE_COLLECTION_FROM_QUERY` method. The query is parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

This procedure uses bulk dynamic SQL to perform the fetch and insert operations into the named collection. Two limitations are imposed by this procedure:

1. The MD5 checksum for the member data is not computed.
2. No column value in query p_query can exceed 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error is raised during execution. In Oracle Database 11g Release 2 (11.2.0.1) or later, this column limit is 4,000 bytes.

Syntax

```
 APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B
(
  p_collection_name IN VARCHAR2,
  p_query IN VARCHAR2,
  p_max_row_count IN NUMBER DEFAULT null);
```

Parameters

Table 5-12 CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version) Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_max_row_count	Maximum number of rows returned from the query in p_query which should be added to the collection.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERY_B procedure to create a collection named EMPLOYEES and populate it with data from the emp table.

```
declare
   l_query varchar2(4000);
Begin
     l_query := 'select empno, ename, job, sal from emp';
     APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B
(
         p_collection_name => 'EMPLOYEES',
         p_query => l_query );
 End;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure (page 5-23)"

5.34 CREATE_COLLECTION_FROM_QUERYB2 Procedure

Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the CREATE_COLLECTION_FROM_QUERY_2 method. The query is parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised. It is identical to the CREATE_COLLECTION_FROM_QUERY_B, however, the first five columns of the SELECT clause must be numeric and the next five columns must be

date. After the date columns, there can be up to 50 character columns in the SELECT clause

This procedure uses bulk dynamic SQL to perform the fetch and insert operations into the named collection. Two limitations are imposed by this procedure:

1. The MD5 checksum for the member data is not computed.
2. No column value in query p_query can exceed 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error is raised during execution. In Oracle Database 11g Release 2 (11.2.0.1) or later, this column limit is 4,000 bytes.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_names IN apex_application_global.vc_arr2,
    p_values IN apex_application_global.vc_arr2,
    p_max_row_count IN NUMBER DEFAULT null);
```

Parameters

Table 5-13 CREATE_COLLECTION_FROM_QUERYB2 Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_names	Array of bind variable names used in the query statement.
p_values	Array of bind variable values used in the bind variables in the query statement.
p_max_row_count	Maximum number of rows returned from the query in p_query which should be added to the collection.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERYB2 procedure to create a collection named EMPLOYEES and populate it with data from the EMP table. The first five columns (mgr, sal, comm, deptno, and null) are all numeric and the next five are all date.

```
declare
 l_query varchar2(4000);
Begin
 l_query := 'select empno, sal, comm, deptno, null, hiredate, null,
null, null, null, ename, job, mgr from emp where deptno = :b1';
  APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
        p_collection_name => 'EMPLOYEES',
        p_query => l_query,
        p_names => apex_util.string_to_table('b1'),
```

```
        p_values => apex_util.string_to_table('10'));
End;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"

5.35 CREATE_COLLECTION_FROM_QUERYB2 Procedure (No bind version)

Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the `CREATE_COLLECTION_FROM_QUERY_2` method. The query is parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised. It is identical to the `CREATE_COLLECTION_FROM_QUERY_B`, however, the first five columns of the `SELECT` clause must be numeric and the next five columns must be date. After the date columns, there can be up to 50 character columns in the `SELECT` clause

This procedure uses bulk dynamic SQL to perform the fetch and insert operations into the named collection. Two limitations are imposed by this procedure:

1. The MD5 checksum for the member data is not computed.
2. No column value in query p_query can exceed 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error is raised during execution. In Oracle Database 11g Release 2 (11.2.0.1) or later, this column limit is 4,000 bytes.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2
(
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2.
    p_max_row_count IN NUMBER DEFAULT);
```

Parameters

Table 5-14 CREATE_COLLECTION_FROM_QUERYB2 Procedure (No bind version) Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_max_row_count	Maximum number of rows returned from the query in p_query which should be added to the collection.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERYB2 procedure to create a collection named EMPLOYEES and populate it with data from the EMP table. The first five columns (mgr, sal, comm, deptno, and null) are all numeric and the next five are all date. Because p_generate_md5 is 'NO', the MD5 checksum is not computed.

```
declare
  l_query varchar2(4000);
Begin
 l_query := 'select empno, sal, comm, deptno, null, hiredate, null,
null, null, null, ename, job, mgr from emp where deptno = 10';
  APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2
(
        p_collection_name => 'EMPLOYEES',
        p_query => l_query,
End;
```

See Also:

- "GET_MEMBER_MD5 Function (page 5-30)"
- "CREATE_COLLECTION Procedure (page 5-16)"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"
- "CREATE_COLLECTION_FROM_QUERY Procedure (page 5-18)"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure (page 5-19)"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure (page 5-21)"

5.36 DELETE_ALL_COLLECTIONS Procedure

Use this procedure to delete all collections that belong to the current user in the current Application Express session for the current Application ID.

Syntax

```
APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
```

Parameters

None.

Example

This example shows how to use the `DELETE_ALL_COLLECTIONS` procedure to remove all collections that belong to the current user in the current session and Application ID.

```
Begin
    APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
End;
```

See Also:

- "DELETE_ALL_COLLECTIONS Procedure (page 5-26),"
- "DELETE_COLLECTION Procedure (page 5-28)"
- "DELETE_MEMBER Procedure (page 5-29)"
- "DELETE_MEMBERS Procedure (page 5-29)"

5.37 DELETE_ALL_COLLECTIONS_SESSION Procedure

Use this procedure to delete all collections that belong to the current user in the current Application Express session regardless of the Application ID.

Syntax

```
APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
```

Parameters

None.

Example

This example shows how to use the `DELETE_ALL_COLLECTIONS_SESSION` procedure to remove all collections that belong to the current user in the current session regardless of Application ID.

```
Begin
    APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
End;
```

See Also:

- "DELETE_ALL_COLLECTIONS Procedure (page 5-26)"
- "DELETE_COLLECTION Procedure (page 5-28)"
- "DELETE_MEMBER Procedure (page 5-29)"
- "DELETE_MEMBERS Procedure (page 5-29)"

5.38 DELETE_COLLECTION Procedure

Use this procedure to delete a named collection. All members that belong to the collection are removed and the named collection is dropped. If the named collection does not exist for the same user in the current session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.DELETE_COLLECTION (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 5-15 DELETE_COLLECTION Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection to remove all members from and drop. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to use the DELETE_COLLECTION procedure to remove the 'EMPLOYEE' collection.

```
Begin
    APEX_COLLECTION.DELETE_COLLECTION(
        p_collection_name => 'EMPLOYEE');
End;
```

See Also:

- "DELETE_ALL_COLLECTIONS_SESSION Procedure (page 5-27)"
- "DELETE_ALL_COLLECTIONS Procedure (page 5-26)"
- "DELETE_MEMBER Procedure (page 5-29)"
- "DELETE_MEMBERS Procedure (page 5-29)"

5.39 DELETE_MEMBER Procedure

Use this procedure to delete a specified member from a given named collection. If the named collection does not exist for the same user in the current session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.DELETE_MEMBER (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2);
```

Parameters

Table 5-16 DELETE_MEMBER Parameters

Parameter	Description
p_collection_name	The name of the collection to delete the specified member from. The maximum length is 255 characters. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist for the current user in the same session.
p_seq	This is the sequence ID of the collection member to be deleted.

Example

This example shows how to use the DELETE_MEMBER procedure to remove the member with a sequence ID of '2' from the collection named EMPLOYEES.

```
Begin
    APEX_COLLECTION.DELETE_MEMBER(
        p_collection_name => 'EMPLOYEES',
        p_seq => '2');
End;
```

See Also:

- "DELETE_ALL_COLLECTIONS_SESSION Procedure (page 5-27)"
- "DELETE_ALL_COLLECTIONS Procedure (page 5-26)"
- "DELETE_COLLECTION Procedure (page 5-28)"
- "DELETE_MEMBERS Procedure (page 5-29)"

5.40 DELETE_MEMBERS Procedure

Use this procedure to delete all members from a given named collection where the attribute specified by the attribute number equals the supplied value. If the named collection does not exist for the same user in the current session for the current Application ID, an application error is raised. If the attribute number specified is invalid or outside the range of 1 to 50, an error is raised.

If the supplied attribute value is null, then all members of the named collection are deleted where the attribute, specified by `p_attr_number`, is null.

Syntax

```
APEX_COLLECTION.DELETE_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_attr_number IN VARCHAR2,
    p_attr_value IN VARCHAR2);
```

Parameters

Table 5-17 DELETE_MEMBERS Parameters

Parameter	Description
`p_collection_name`	The name of the collection to delete the specified members from. The maximum length is 255 characters. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist for the current user in the same session.
`p_attr_number`	Attribute number of the member attribute used to match for the specified attribute value for deletion. Valid values are 1 through 50 and null.
`p_attr_value`	Attribute value of the member attribute used to match for deletion. Maximum length can be 4,000 bytes. The attribute value is truncated to 4,000 bytes if greater than this amount.

Example

The following example deletes all members of the collection named 'GROCERIES' where the 5th character attribute is equal to 'APPLE'.

```
Begin
    apex_collection.delete_members(
        p_collection_name => 'GROCERIES'
        p_attr_number     => 5,
        p_attr_value      => 'APPLE' );
    Commit;
End;
```

See Also:

- "DELETE_ALL_COLLECTIONS_SESSION Procedure (page 5-27)"
- "DELETE_ALL_COLLECTIONS Procedure (page 5-26)"
- "DELETE_COLLECTION Procedure (page 5-28)"
- "DELETE_MEMBER Procedure (page 5-29)"

5.41 GET_MEMBER_MD5 Function

Use this function to compute and return the message digest of the attributes for the member specified by the sequence ID. This computation of message digest is equal to

the computation performed natively by collections. Thus, the result of this function could be compared to the MD5_ORIGINAL column of the view apex_collections.

If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised.

Syntax

```
APEX_COLLECTION.GET_MEMBER_MD5 (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 5-18 GET_MEMBER_MD5 Parameters

Parameter	Description
p_collection_name	The name of the collection to add this array of members to. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
p_seq	Sequence ID of the collection member.

Example

The following example computes the MD5 for the 5th member of the GROCERIES collection.

```
declare
    l_md5 varchar2(4000);
begin
    l_md5 := apex_collection.get_member_md5(
        p_collection_name => 'GROCERIES'
        p_seq             => 10 );
end;
```

See Also:

- "COLLECTION_HAS_CHANGED Function (page 5-15)"
- "RESET_COLLECTION_CHANGED Procedure (page 5-36)"
- "RESET_COLLECTION_CHANGED_ALL Procedure (page 5-36)"

5.42 MERGE_MEMBERS Procedure

Use this procedure to merge members of the given named collection with the values passed in the arrays. If the named collection does not exist one is created. If a p_init_query is provided, the collection is created from the supplied SQL query. If the named collection exists, the following occurs:

1. Rows in the collection and not in the arrays are deleted.
2. Rows in the collections and in the arrays are updated.

3. Rows in the arrays and not in the collection are inserted.

The count of elements in the p_c001 PL/SQL table is used as the total number of items across all PL/SQL tables. For example, if p_c001.count is 2 and p_c002.count is 10, only 2 members are merged. If p_c001 is null an application error is raised.

Syntax

```
APEX_COLLECTION.MERGE_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_seq  IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_c001 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_c002 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_c003 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    ...
    p_c050 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_null_index  IN NUMBER DEFAULT 1,
    p_null_value  IN VARCHAR2 DEFAULT null,
    p_init_query  IN VARCHAR2 DEFAULT null);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-19 MERGE_MEMBERS Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
p_c001 through p_c050	Array of attribute values to be merged. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. The count of the p_c001 array is used across all arrays. If no values are provided then no actions are performed.
p_c0xx	Attribute of NN attributes values to be merged. Maximum length can be 4,000 bytes. The attribute value is truncated to 4,000 bytes if greater than this amount.
p_seq	Identifies the sequence number of the collection to be merged.
p_null_index	That is if the element identified by this value is null, then treat this row as a null row. For example, if p_null_index is 3, then p_c003 is treated as a null row. In other words, tell the merge function to ignore this row. This results in the null rows being removed from the collection. The null index works with the null value. If the value of the p_cXXX argument is equal to the p_null_value then the row is treated as null.

Table 5-19 (Cont.) MERGE_MEMBERS Procedure Parameters

Parameter	Description
p_null_value	Used with the p_null_index argument. Identifies the null value. If used, this value must not be null. A typical value for this argument is "0"
p_init_query	If the collection does not exist, the collection is created using this query.

Example

The following example creates a collection on the table of employees, and then merges the contents of the local arrays with the collection, updating the job of two employees.

```
DECLARE
    l_seq   APEX_APPLICATION_GLOBAL.VC_ARR2;
    l_c001  APEX_APPLICATION_GLOBAL.VC_ARR2;
    l_c002  APEX_APPLICATION_GLOBAL.VC_ARR2;
    l_c003  APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_seq(1)  := 1;
    l_c001(1) := 7369;
    l_c002(1) := 'SMITH';
    l_c003(1) := 'MANAGER';
    l_seq(2)  := 2;
    l_c001(2) := 7499;
    l_c002(2) := 'ALLEN';
    l_c003(2) := 'CLERK';

    APEX_COLLECTION.MERGE_MEMBERS(
        p_collection_name => 'EMPLOYEES',
        p_seq => l_seq,
        p_c001 => l_c001,
        p_c002 => l_c002,
        p_c003 => l_c003,
        p_init_query => 'select empno, ename, job from emp order by empno');
END;
```

5.43 MOVE_MEMBER_DOWN Procedure

Use this procedure to adjust the sequence ID of specified member in the given named collection down by one (subtract one), swapping sequence ID with the one it is replacing. For example, 3 becomes 2 and 2 becomes 3. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the member specified by sequence ID p_seq is the lowest sequence in the collection, an application error is NOT returned.

Syntax

```
APEX_COLLECTION.MOVE_MEMBER_DOWN (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER);
```

Parameters

Table 5-20 MOVE_MEMBER_DOWN Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist with the specified name of the current user in the same session.
p_seq	Identifies the sequence number of the collection member to be moved down by one.

Example

This example shows how to a member of the EMPLOYEES collection down one position. After executing this example, sequence ID '5' becomes sequence ID '4' and sequence ID '4' becomes sequence ID '5'.

```
BEGIN;
    APEX_COLLECTION.MOVE_MEMBER_DOWN(
        p_collection_name => 'EMPLOYEES',
        p_seq => '5' );
END;
```

See Also:

"MOVE_MEMBER_UP Procedure (page 5-34)"

5.44 MOVE_MEMBER_UP Procedure

Use this procedure to adjust the sequence ID of specified member in the given named collection up by one (add one), swapping sequence ID with the one it is replacing. For example, 2 becomes 3 and 3 becomes 2. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the member specified by sequence ID p_seq is the highest sequence in the collection, an application error is not returned.

Syntax

```
APEX_COLLECTION.MOVE_MEMBER_UP (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER);
```

Parameters

Table 5-21 MOVE_MEMBER_UP Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist with the specified name of the current user in the same session.

Table 5-21 (Cont.) MOVE_MEMBER_UP Parameters

Parameter	Description
p_seq	Identifies the sequence number of the collection member to be moved up by one.

Example

This example shows how to a member of the EMPLOYEES collection down one position. After executing this example, sequence ID '5' becomes sequence ID '6' and sequence ID '6' becomes sequence ID '5'.

```
BEGIN;
    APEX_COLLECTION.MOVE_MEMBER_UP(
        p_collection_name => 'EMPLOYEES',
        p_seq => '5' );
END;
```

See Also:

"MOVE_MEMBER_DOWN Procedure (page 5-33)"

5.45 RESEQUENCE_COLLECTION Procedure

For a named collection, use this procedure to update the seq_id value of each member so that no gaps exist in the sequencing. For example, a collection with the following set of sequence IDs (1,2,3,5,8,9) becomes (1,2,3,4,5,6). If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.RESEQUENCE_COLLECTION (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 5-22 RESEQUENCE_COLLECTION Parameters

Parameter	Description
p_collection_name	The name of the collection to resequence. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to resequence the DEPARTMENTS collection to remove gaps in the sequence IDs.

```
BEGIN;
    APEX_COLLECTION.RESEQUENCE_COLLECTION (
        p_collection_name => 'DEPARTMENTS');
END;
```

See Also:

- "MOVE_MEMBER_DOWN Procedure (page 5-33)"
- "MOVE_MEMBER_UP Procedure (page 5-34)"

5.46 RESET_COLLECTION_CHANGED Procedure

Use this procedure to reset the collection changed flag (mark as not changed) for a given collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.RESET_COLLECTION_CHANGED (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 5-23 RESET_COLLECTION_CHANGED Parameters

Parameter	Description
p_collection_name	The name of the collection to reset the collection changed flag. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to reset the changed flag for the DEPARTMENTS collection.

```
BEGIN;
    APEX_COLLECTION.RESET_COLLECTION_CHANGED (
        p_collection_name => 'DEPARTMENTS');
END;
```

See Also:

"RESET_COLLECTION_CHANGED_ALL Procedure (page 5-36)"

5.47 RESET_COLLECTION_CHANGED_ALL Procedure

Use this procedure to reset the collection changed flag (mark as not changed) for all collections in the user's current session.

Syntax

```
APEX_COLLECTION.RESET_COLLECTION_CHANGED_ALL; (
```

Parameters

None.

Example

This example shows how to reset the changed flag for all collections in the user's current session.

```
BEGIN;
    APEX_COLLECTION.RESET_COLLECTION_CHANGED_ALL;
END;
```

See Also:

"RESET_COLLECTION_CHANGED Procedure (page 5-36)"

5.48 SORT_MEMBERS Procedure

Use this procedure to reorder the members of a given collection by the column number specified by p_sort_on_column_number. This sorts the collection by a particular column/attribute in the collection and reassigns the sequence IDs of each number such that no gaps exist. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.SORT_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_sort_on_column_number IN NUMBER);
```

Parameters

Table 5-24 SORT_MEMBERS Parameters

Parameter	Description
p_collection_name	The name of the collection to sort. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
p_sort_on_column_number	The column number used to sort the collection. The domain of possible values is 1 to 50.

Example

In this example, column 2 of the DEPARTMENTS collection is the department location. The collection is reorder according to the department location.

```
BEGIN;
    APEX_COLLECTION.SORT_MEMBERS (
        p_collection_name => 'DEPARTMENTS',
        p_sort_on_column_number => '2';
END;
```

5.49 TRUNCATE_COLLECTION Procedure

Use this procedure to remove all members from a named collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.TRUNCATE_COLLECTION (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 5-25 TRUNCATE_COLLECTION Parameters

Parameter	Description
p_collection_name	The name of the collection to truncate. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to remove all members from the DEPARTMENTS collection.

```
BEGIN;
    APEX_COLLECTION.TRUNCATE_COLLECTION(
        p_collection_name => 'DEPARTMENTS');
END;
```

See Also:

"CREATE_OR_TRUNCATE_COLLECTION Procedure (page 5-17)"

5.50 UPDATE_MEMBER Procedure

Use this procedure to update the specified member in the given named collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised.

Note:

Using this procedure sets the columns identified and nullifies any columns not identified. To update specific columns, without affecting the values of other columns, use " UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1".

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER (
    p_collection_name IN VARCHAR2,
    p_seq  IN VARCHAR2 DEFAULT NULL,
    p_c001 IN VARCHAR2 DEFAULT NULL,
    p_c002 IN VARCHAR2 DEFAULT NULL,
    p_c003 IN VARCHAR2 DEFAULT NULL,
    ...
    p_c050 IN VARCHAR DEFAULT NULL,
    p_n001 IN NUMBER DEFAULT NULL,
    p_n002 IN NUMBER DEFAULT NULL,
    p_n003 IN NUMBER DEFAULT NULL,
```

```
    p_n004 IN NUMBER DEFAULT NULL,
    p_n005 IN NUMBER DEFAULT NULL,
    p_d001 IN DATE DEFAULT NULL,
    p_d002 IN DATE DEFAULT NULL,
    p_d003 IN DATE DEFAULT NULL,
    p_d004 IN DATE DEFAULT NULL,
    p_d005 IN DATE DEFAULT NULL,
    p_clob001 IN CLOB DEFAULT empty_clob(),
    p_blob001 IN BLOB DEFAULT empty-blob(),
    p_xmltype001 IN XMLTYPE DEFAULT NULL);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-26 UPDATE_MEMBER Parameters

Parameter	Description
`p_collection_name`	The name of the collection to update. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
`p_n001 through p_n005`	Attribute value of the numeric attributes to be added or updated.
`p_d001` through `p_d005`	Attribute value of the date attributes to be added or updated.
`p_clob001`	Use p_clob001 for collection member attributes that exceed 4,000 characters.
`p_blob001`	Use p_blob001 for binary collection member attributes.
`p_xmltype001`	Use p_xmltype001 to store well-formed XML.

Example

Update the second member of the collection named 'Departments', updating the first member attribute to 'Engineering' and the second member attribute to 'Sales'.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER (
        p_collection_name => 'Departments',
        p_seq => '2',
        p_c001 => 'Engineering',
        p_c002 => 'Sales');
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBERS Procedure (page 5-40)"

5.51 UPDATE_MEMBERS Procedure

Use this procedure to update the array of members for the given named collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error is raised. The count of elements in the p_seq PL/SQL table is used as the total number of items across all PL/SQL tables. That is, if p_seq.count = 2 and p_c001.count = 10, only 2 members are updaapex_application_globalted. If p_seq is null, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_seq  IN apex_application_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_c001 IN apex_application_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_c002 IN apex_application_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_c003 IN apex_application_global.VC_ARR2 DEFAULT empty_vc_arr,
    ...
    p_c050 IN apex_application_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_n001 IN apex_application_global.N_ARR DEFAULT empty_n_arr,
    p_n002 IN apex_application_global.N_ARR DEFAULT empty_n_arr,
    p_n003 IN apex_application_global.N_ARR DEFAULT empty_n_arr,
    p_n004 IN apex_application_global.N_ARR DEFAULT empty_n_arr,
    p_n005 IN apex_application_global.N_ARR DEFAULT empty_n_arr,
    p_d001 IN apex_application_global.D_ARR DEFAULT empty_d_arr,
    p_d002 IN apex_application_global.D_ARR DEFAULT empty_d_arr,
    p_d003 IN apex_application_global.D_ARR DEFAULT empty_d_arr,
    p_d004 IN apex_application_global.D_ARR DEFAULT empty_d_arr,
    p_d005 IN apex_application_global.D_ARR DEFAULT empty_d_arr)
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-27 UPDATE_MEMBERS Parameters

Parameter	Description
p_collection_name	The name of the collection to update. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.

Table 5-27 (Cont.) UPDATE_MEMBERS Parameters

Parameter	Description
p_seq	Array of member sequence IDs to be updated. The count of the p_seq array is used across all arrays.
p_c001 through p_c050	Array of attribute values to be updated.
p_n001 through p_n005	Attribute value of numeric
p_d001 through p_d005	Array of date attribute values to be updated.

Example

```
DECLARE
    l_seq   apex_application_global.vc_arr2;
    l_carr  apex_application_global.vc_arr2;
    l_narr  apex_application_global.n_arr;
    l_darr  apex_application_global.d_arr;
BEGIN
    l_seq(1)  := 10;
    l_seq(2)  := 15;
    l_carr(1) := 'Apples';
    l_carr(2) := 'Grapes';
    l_narr(1) := 100;
    l_narr(2) := 150;
    l_darr(1) := sysdate;
    l_darr(2) := sysdate;

    APEX_COLLECTION.UPDATE_MEMBERS (
        p_collection_name => 'Groceries',
        p_seq  => l_seq,
        p_c001 => l_carr,
        p_n001 => l_narr,
        p_d001 => l_darr);
END;
```

See Also:

"UPDATE_MEMBER Procedure (page 5-38)"

5.52 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1

Update the specified member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the range 1-50, an error is raised. Any attribute value exceeding 4,000 bytes are truncated to 4,000 bytes.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER,
    p_attr_number IN NUMBER,
    p_attr_value  IN VARCHAR2);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-28 UPDATE_MEMBER_ATTRIBUTE Signature 1 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_attr_number`	Attribute number of the member attribute to be updated. Valid values are 1 through 50. Any number outside of this range is ignored.
`p_attr_value`	Attribute value of the member attribute to be updated.

Example

Update the second member of the collection named `'Departments'`, updating the first member attribute to `'Engineering'` and the second member attribute to `'Sales'`.

```
BEGIN
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name => 'Departments',
        p_seq => 2,
        p_attr_number => 1,
        p_attr_value => 'Engineering');
END;
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 (page 5-43)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 (page 5-44)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 (page 5-45)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 (page 5-47)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6 (page 5-48)"

5.53 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2

Update the specified CLOB member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the valid range (currently only 1 for CLOB), an error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER,
    p_clob_number IN NUMBER,
    p_clob_value  IN CLOB);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-29 UPDATE_MEMBER_ATTRIBUTE Signature 2 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_clob_number`	Attribute number of the CLOB member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.
`p_clob_value`	Attribute value of the CLOB member attribute to be updated.

Example

The following example sets the first and only CLOB attribute of collection sequence number 2 in the collection named `'Departments'` to a value of `'Engineering'`.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name => 'Departments',
        p_seq => 2,
        p_clob_number => 1,
        p_clob_value => 'Engineering');
END;
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 (page 5-44)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 (page 5-45)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 (page 5-47)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6 (page 5-48)"

5.54 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3

Update the specified BLOB member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the valid range (currently only 1 for BLOB), an error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER,
    p_blob_number IN NUMBER,
    p_blob_value  IN BLOB);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-30 UPDATE_MEMBER_ATTRIBUTE Signature 3 Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
p_seq	Sequence ID of the collection member to be updated.
p_blob_number	Attribute number of the BLOB member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.
p_blob_value	Attribute value of the BLOB member attribute to be updated.

Example

The following example sets the first and only BLOB attribute of collection sequence number 2 in the collection named `'Departments'` to a value of the BLOB variable `l_blob_content`.

```
BEGIN
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name => 'Departments',
        p_seq => 2,
        p_blob_number => 1,
        p_blob_value => l_blob_content);
END;
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 (page 5-43)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 (page 5-45)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 (page 5-47)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6 (page 5-48)"

5.55 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4

Update the specified XMLTYPE member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the valid range (currently only 1 for XMLTYPE), an error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER,
    p_xmltype_number IN NUMBER,
    p_xmltype_value  IN BLOB);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-31 UPDATE_MEMBER_ATTRIBUTE Signature 4 Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
p_seq	Sequence ID of the collection member to be updated.
p_xmltype_number	Attribute number of the XMLTYPE member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.
p_xmltype_value	Attribute value of the XMLTYPE member attribute to be updated.

Example

The following example sets the first and only XML attribute of collection sequence number 2 in the collection named `'Departments'` to a value of the XMLType variable `l_xmltype_content`.

```
BEGIN
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name => 'Departments',
        p_seq => 2,
        p_xmltype_number => 1,
        p_xmltype_value => l_xmltype_content);
END;
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 (page 5-43)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 (page 5-44)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 (page 5-47)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6 (page 5-48)"

5.56 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5

Update the specified NUMBER member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for NUMBER), an error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER,
    p_attr_number IN NUMBER,
    p_number_value  IN NUMBER);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-32 UPDATE_MEMBER_ATTRIBUTE Signature 5 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_attr_number`	Attribute number of the NUMBER member attribute to be updated. Valid value is 1 through 5. Any number outside of this range is ignored.

Table 5-32 (Cont.) UPDATE_MEMBER_ATTRIBUTE Signature 5 Parameters

Parameter	Description
p_number_value	Attribute value of the NUMBER member attribute to be updated.

Example

The following example sets the first numeric attribute of collection sequence number 2 in the collection named 'Departments' to a value of 3000.

```
BEGIN
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name => 'Departments',
        p_seq => 2,
        p_attr_number => 1,
        p_number_value => 3000);
END;
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 (page 5-43)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 (page 5-44)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 (page 5-45)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6 (page 5-48)"

5.57 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6

Update the specified DATE member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for DATE), an error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER,
    p_attr_number IN NUMBER,
    p_date_value  IN DATE);
```

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 5-33 UPDATE_MEMBER_ATTRIBUTE Signature 6 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_attr_number`	Attribute number of the DATE member attribute to be updated. Valid value is 1 through 5. Any number outside of this range is ignored.
`p_date_value`	Attribute value of the DATE member attribute to be updated.

Example

Update the first attribute of the second collection member in collection named `'Departments'`, and set it to a value of 100.

```
BEGIN
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name => 'Departments',
        p_seq => 2,
        p_attr_number => 1,
        p_date_value => 100 );
END;
```

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 (page 5-41)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 (page 5-43)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 (page 5-44)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 (page 5-45)"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 (page 5-47)"

6

APEX_CSS

The APEX_CSS package provides utility functions for adding CSS styles to HTTP output. This package is usually used for plug-in development.

6.1 ADD Procedure

This procedure adds a CSS style snippet that is included inline in the HTML output. Use this procedure to add new CSS style declarations.

Syntax

```
APEX_CSS.ADD (
    p_css          IN    VARCHAR2,
    p_key          IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 6-1 ADD Parameters

Parameter	Description
p_css	The CSS style snippet. For example, `#test {color:#fff}`
p_key	Identifier for the style snippet. If specified and a style snippet with the same name has already been added the new style snippet will be ignored.

Example

Adds an inline CSS definition for the class autocomplete into the HTML page. The key `autocomplete_widget` prevents the definition from being included another time if the `apex_css.add` is called another time.

```
apex_css.add (
    p_css => '.autocomplete { color:#ffffff }',
    p_key => 'autocomplete_widget' );
```

6.2 ADD_3RD_PARTY_LIBRARY_FILE Procedure

This procedure adds the link tag to load a 3rd party css file and also takes into account the specified Content Delivery Network for the application. Supported libraries include: `jQuery`, `jQueryUI`, `jQueryMobile`.

If a library has already been added, it is not added a second time.

Syntax

```
add_3rd_party_library_file (
    p_library in varchar2,
    p_file_name in varchar2,
    p_directory in varchar2 default null,
    p_version in varchar2 default null,
    p_media_query in varchar2 default null );
```

Parameters

Table 6-2 ADD_3RD_PARTY_LIBRARY_FILE Parameters

Parameters	Description
`p_library`	Use one of the c_library_* constants
`p_file_name`	Specifies the file name without version, .min and .css
`p_directory`	Directory where the file `p_file_name` is located (optional)
`p_version`	If no value is provided then the same version Application Express ships is used (optional)
`p_media_query`	Value that is set as media query (optional)

Example

The following example loads the Cascading Style Sheet file of the Accordion component of the jQuery UI.

```
apex_css.add_3rd_party_library_file (
    p_library   => apex_css.c_library_jquery_ui,
    p_file_name => 'jquery.ui.accordion' )
```

6.3 ADD_FILE Procedure

This procedure adds the link tag to load a CSS library. If a library has already been added, it will not be added a second time.

Syntax

```
APEX_CSS.ADD_FILE (
    p_name           IN    VARCHAR2,
    p_directory      IN    VARCHAR2 DEFAULT APEX.G_IMAGE_PREFIX||'css/',
    p_version        IN    VARCHAR2 DEFAULT NULL,
    p_skip_extension IN    BOOLEAN DEFAULT FALSE
    p_media_query    IN    VARCHAR2 DEFAULT NULL,
    p_ie_condition   IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 6-3 ADD_FILE Parameters

Parameter	Description
p_name	Name of the CSS file.
p_directory	Begin of the URL where the CSS file should be read from. If you use this function for a plug-in you should set this parameter to p_plugin.file_prefix.
p_version	Identifier of the version of the CSS file. The version will be added to the CSS filename. In most cases you should use the default of NULL as the value.
p_skip_extension	The function automatically adds ".css" to the CSS filename. If this parameter is set to TRUE this will not be done.
p_media_query	Value set as media query.
p_ie_condition	Condition used as Internet Explorer condition.

Example

Adds the CSS filejquery.autocomplete.css in the directory specified by p_plugin.image_prefix to the HTML output of the page and makes sure that it will only be included once if apex_css.add_file is called multiple times with that name.

```
apex_css.add_file (
    p_name => 'jquery.autocomplete',
    p_directory => p_plugin.image_prefix );
```

7

APEX_CUSTOM_AUTH

You can use the `APEX_CUSTOM_AUTH` package to perform various operations related to authentication and session management.

7.1 APPLICATION_PAGE_ITEM_EXISTS Function

This function checks for the existence of page-level item within the current page of an application. This function requires the parameter `p_item_name`. This function returns a Boolean value (TRUE or FALSE).

Syntax

```
APEX_CUSTOM_AUTH.APPLICATION_PAGE_ITEM_EXISTS(
    p_item_name   IN    VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 7-1 APPLICATION_PAGE_ITEM_EXISTS Parameters

Parameter	Description
p_item_name	The name of the page-level item.

Example

The following example checks for the existence of a page-level item, `ITEM_NAME`, within the current page of the application.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
   L_VAL := APEX_CUSTOM_AUTH.APPLICATION_PAGE_ITEM_EXISTS(:ITEM_NAME);
    IF L_VAL THEN
        htp.p('Item Exists');
    ELSE
        htp.p('Does not Exist');
    END IF;
END;
```

7.2 CURRENT_PAGE_IS_PUBLIC Function

This function checks whether the current page's authentication attribute is set to **Page Is Public** and returns a Boolean value (TRUE or FALSE)

Syntax

```
APEX_CUSTOM_AUTH.CURRENT_PAGE_IS_PUBLIC
RETURN BOOLEAN;
```

Example

The following example checks whether the current page in an application is public.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    L_VAL := APEX_CUSTOM_AUTH.CURRENT_PAGE_IS_PUBLIC;
    IF L_VAL THEN
        htp.p('Page is Public');
    ELSE
        htp.p('Page is not Public');
    END IF;
END;
```

See Also:

"Editing Page Attributes" in *Oracle Application Express App Builder User's Guide*.

7.3 DEFINE_USER_SESSION Procedure

This procedure combines the `SET_USER` and `SET_SESSION_ID` procedures to create one call.

Syntax

```
APEX_CUSTOM_AUTH.DEFINE_USER_SESSION(
    p_user         IN    VARCHAR2,
    p_session_id   IN    NUMBER);
```

Parameters

Table 7-2 DEFINE_USER_SESSION Parameters

Parameter	Description
`p_user`	Login name of the user.
`p_session_id`	The session ID.

Example

In the following example, a new session ID is generated and registered along with the current application user.

```
APEX_CUSTOM_AUTH.DEFINE_USER_SESSION (
    :APP_USER,
    APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID);
```

See Also:

- "SET_USER Procedure (page 7-11)"
- "SET_SESSION_ID Procedure (page 7-11)"

7.4 GET_COOKIE_PROPS Procedure

This procedure obtains the properties of the session cookie used in the current authentication scheme for the specified application. These properties can be viewed directly in the App Builder by viewing the authentication scheme cookie attributes.

Syntax

```
APEX_CUSTOM_AUTH.GET_COOKIE_PROPS(
    p_app_id                    IN  NUMBER,
    p_cookie_name               OUT VARCHAR2,
    p_cookie_path               OUT VARCHAR2,
    p_cookie_domain             OUT VARCHAR2
    p_secure                            OUT BOOLEAN);
```

Parameters

Table 7-3 GET_COOKIE_PROPS Parameters

Parameter	Description
p_app_id	An application ID in the current workspace.
p_cookie_name	The cookie name.
p_cookie_path	The cookie path.
p_cookie_domain	The cookie domain.
p_secure	Flag to set secure property of cookie.

Example

The following example retrieves the session cookie values used by the authentication scheme of the current application.

```
DECLARE
    l_cookie_name   varchar2(256);
    l_cookie_path   varchar2(256);
    l_cookie_domain varchar2(256);
    l_secure        boolean;
BEGIN
    APEX_CUSTOM_AUTH.GET_COOKIE_PROPS(
        p_app_id => 2918,
        p_cookie_name => l_cookie_name,
        p_cookie_path => l_cookie_path,
        p_cookie_domain => l_cookie_domain,
        p_secure => l_secure);
END;
```

7.5 GET_LDAP_PROPS Procedure

This procedure obtains the LDAP attributes of the current authentication scheme for the current application. These properties can be viewed directly in App Builder by viewing the authentication scheme attributes.

Syntax

```
APEX_CUSTOM_AUTH.GET_LDAP_PROPS(
    p_ldap_host               OUT VARCHAR2,
    p_ldap_port               OUT INTEGER,
    p_use_ssl                 OUT VARCHAR2,
    p_use_exact_dn            OUT VARCHAR2,
    p_search_filter           OUT VARCHAR2,
    p_ldap_dn                 OUT VARCHAR2,
    p_ldap_edit_function      OUT VARCHAR2);
```

Parameters

Table 7-4 GET_LDAP_PROPS Parameters

Parameter	Description
p_ldap_host	LDAP host name.
p_ldap_port	LDAP port number.
p_use_ssl	Whether SSL is used.
p_use_exact_dn	Whether exact distinguished names are used.
p_search_filter	The search filter used if exact DN is not used.
p_ldap_dn	LDAP DN string.
p_ldap_edit_function	LDAP edit function name.

Example

The following example retrieves the LDAP attributes associated with the current application.

```
DECLARE
    l_ldap_host          VARCHAR2(256);
    l_ldap_port          INTEGER;
    l_use_ssl            VARCHAR2(1);
    l_use_exact_dn       VARCHAR2(1);
    l_search_filter      VARCHAR2(256);
    l_ldap_dn            VARCHAR2(256);
    l_ldap_edit_function VARCHAR2(256);
BEGIN
APEX_CUSTOM_AUTH.GET_LDAP_PROPS (
    p_ldap_host       => l_ldap_host,
    p_ldap_port       => l_ldap_port,
    p_use_ssl         => l_use_ssl,
    p_use_exact_dn    => l_use_exact_dn,
    p_search_filter   => l_search_filter,
    p_ldap_dn         => l_ldap_dn,
    p_ldap_edit_function => l_ldap_edit_function);
END;
```

7.6 GET_NEXT_SESSION_ID Function

This function generates the next session ID from the Oracle Application Express sequence generator. This function returns a number.

Syntax

```
APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID
RETURN NUMBER;
```

Example

The following example generates the next session ID and stores it into a variable.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID;
END;
```

7.7 GET_SECURITY_GROUP_ID Function

This function returns a number with the value of the security group ID that identifies the workspace of the current user.

Syntax

```
APEX_CUSTOM_AUTH.GET_SECURITY_GROUP_ID
RETURN NUMBER;
```

Example

The following example retrieves the Security Group ID for the current user.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_SECURITY_GROUP_ID;
END;
```

7.8 GET_SESSION_ID Function

This function returns `APEX_APPLICATION.G_INSTANCE` global variable. `GET_SESSION_ID` returns a number.

Syntax

```
APEX_CUSTOM_AUTH.GET_SESSION_ID
RETURN NUMBER;
```

Example

The following example retrieves the session ID for the current user.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_SESSION_ID;
END;
```

7.9 GET_SESSION_ID_FROM_COOKIE Function

This function returns the Oracle Application Express session ID located by the session cookie in a page request in the current browser session.

Syntax

```
APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE
RETURN NUMBER;
```

Example

The following example retrieves the session ID from the current session cookie.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE;
END;
```

7.10 GET_USER Function

This function returns the APEX_APPLICATION.G_USER global variable (VARCHAR2).

Syntax

```
APEX_CUSTOM_AUTH.GET_USER
RETURN VARCHAR2;
```

Examples

The following example retrieves the username associated with the current session.

```
DECLARE
    VAL VARCHAR2(256);
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_USER;
END;
```

7.11 GET_USERNAME Function

This function returns user name registered with the current Oracle Application Express session in the internal sessions table. This user name is usually the same as the authenticated user running the current page.

Syntax

```
APEX_CUSTOM_AUTH.GET_USERNAME
RETURN VARCHAR2;
```

Example

The following example retrieves the username registered with the current application session.

```
DECLARE
    VAL VARCHAR2(256);
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_USERNAME;
END;
```

7.12 IS_SESSION_VALID Function

This function is a Boolean result obtained from executing the current application's authentication scheme to determine if a valid session exists. This function returns the Boolean result of the authentication scheme's page sentry.

Syntax

```
APEX_CUSTOM_AUTH.IS_SESSION_VALID
RETURN BOOLEAN;
```

Example

The following example verifies whether the current session is valid.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    L_VAL := APEX_CUSTOM_AUTH.IS_SESSION_VALID;
    IF L_VAL THEN
        htp.p('Valid');
    ELSE
        htp.p('Invalid');
    END IF;
END;
```

7.13 LOGIN Procedure

Also referred to as the "Login API," this procedure performs authentication and session registration.

Syntax

```
APEX_CUSTOM_AUTH.LOGIN(
    p_uname                     IN  VARCHAR2  DEFAULT NULL,
    p_password                  IN  VARCHAR2  DEFAULT NULL,
    p_session_id                IN  VARCHAR2  DEFAULT NULL,
    p_app_page                  IN  VARCHAR2  DEFAULT NULL,
    p_entry_point               IN  VARCHAR2  DEFAULT NULL,
    p_preserve_case             IN  BOOLEAN   DEFAULT FALSE);
```

Parameter

Table 7-5 LOGIN Parameters

Parameter	Description
p_uname	Login name of the user.
p_password	Clear text user password.
p_session_id	Current Oracle Application Express session ID.
p_app_page	Current application ID. After login page separated by a colon (:).
p_entry_point	Internal use only.
p_preserve_case	If TRUE, do not upper p_uname during session registration

Example

The following example performs the user authentication and session registration.

```
BEGIN
    APEX_CUSTOM_AUTH.LOGIN (
        p_uname       => 'FRANK',
        p_password    => 'secret99',
        p_session_id  => V('APP_SESSION'),
```

```
        p_app_page    => :APP_ID||':1');
END;
```

Note:

Do not use bind variable notations for `p_session_id` argument.

7.14 LOGOUT Procedure [DEPRECATED]

Note:

This procedure is deprecated. Use `APEX_AUTHENTICATION.LOGOUT` instead.

This procedure causes a logout from the current session by unsetting the session cookie and redirecting to a new location.

Syntax

```
APEX_CUSTOM_AUTH.LOGOUT(
    p_this_app                    IN VARCHAR2  DEFAULT NULL,
    p_next_app_page_sess          IN VARCHAR2  DEFAULT NULL,
    p_next_url                    IN VARCHAR2  DEFAULT NULL);
```

Parameter

Table 7-6 LOGOUT Parameters

Parameter	Description
`p_this_app`	Current application ID.
`p_next_app_page_sess`	Application and page number to redirect to. Separate multiple pages using a colon (:) and optionally followed by a colon (:) and the session ID (if control over the session ID is desired).
`p_next_url`	URL to redirect to (use this instead of `p_next_app_page_sess`).

Example

The following example causes a logout from the current session and redirects to page `99` of application `1000`.

```
BEGIN
    APEX_CUSTOM_AUTH.LOGOUT (
        p_this_app            => '1000',
        p_next_app_page_sess  => '1000:99');
END;
```

7.15 POST_LOGIN Procedure

This procedure performs session registration, assuming the authentication step has been completed. It can be called only from within an Oracle Application Express application page context.

Syntax

```
APEX_CUSTOM_AUTH.POST_LOGIN(
    p_uname                    IN  VARCHAR2  DEFAULT NULL,
    p_session_id               IN  VARCHAR2  DEFAULT NULL,
    p_app_page                 IN  VARCHAR2  DEFAULT NULL,
    p_preserve_case            IN  BOOLEAN   DEFAULT FALSE);
```

Parameter

Table 7-7 POST_LOGIN Parameters

Parameter	Description
p_uname	Login name of user.
p_session_id	Current Oracle Application Express session ID.
p_app_page	Current application ID and after login page separated by a colon (:).
p_preserve_case	If TRUE, do not include p_uname in uppercase during session registration.

Example

The following example performs the session registration following a successful authentication.

```
BEGIN
    APEX_CUSTOM_AUTH.POST_LOGIN (
        p_uname       => 'FRANK',
        p_session_id  => V('APP_SESSION'),
        p_app_page    => :APP_ID||':1');
END;
```

7.16 SESSION_ID_EXISTS Function

This function returns a Boolean result based on the global package variable containing the current Oracle Application Express session ID. Returns TRUE if the result is a positive number and returns FALSE if the result is a negative number.

Syntax

```
APEX_CUSTOM_AUTH.SESSION_ID_EXISTS
RETURN BOOLEAN;
```

Example

The following example checks whether the current session ID is valid and exists.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    L_VAL := APEX_CUSTOM_AUTH.SESSION_ID_EXISTS;
    IF L_VAL THEN
        htp.p('Exists');
    ELSE
        htp.p('Does not exist');
```

```
    END IF;
END;
```

7.17 SET_SESSION_ID Procedure

This procedure sets APEX_APPLICATION.G_INSTANCE global variable. This procedure requires the parameter P_SESSION_ID (NUMBER) which specifies a session ID.

Syntax

```
APEX_CUSTOM_AUTH.SET_SESSION_ID(
    p_session_id    IN    NUMBER);
```

Parameters

Table 7-8 SET_SESSION_ID Parameters

Parameter	Description
p_session_id	The session ID to be registered.

Example

In the following example, the session ID value registered is retrieved from the browser cookie.

```
APEX_CUSTOM_AUTH.SET_SESSION_ID(APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE);
```

7.18 SET_SESSION_ID_TO_NEXT_VALUE Procedure

This procedure combines the operation of GET_NEXT_SESSION_ID and SET_SESSION_ID in one call.

Syntax

```
APEX_CUSTOM_AUTH.SET_SESSION_ID_TO_NEXT_VALUE;
```

Example

In the following example, if the current session is not valid, a new session ID is generated and registered.

```
IF NOT APEX_CUSTOM_AUTH.SESSION_ID_EXISTS THEN
    APEX_CUSTOM_AUTH.SET_SESSION_ID_TO_NEXT_VALUE;
END IF;
```

7.19 SET_USER Procedure

This procedure sets the APEX_APPLICATION.G_USER global variable. SET_USER requires the parameter P_USER (VARCHAR2) which defines a user ID.

Syntax

```
APEX_CUSTOM_AUTH.SET_USER(
    p_user   IN    VARCHAR2);
```

Parameters

Table 7-9 SET_USER Parameters

Parameter	Description
p_user	The user ID to be registered.

Example

In the following example, if the current application user is **NOBODY**, then **JOHN.DOE** is registered as the application user.

```
IF V('APP_USER') = 'NOBODY' THEN
    APEX_CUSTOM_AUTH.SET_USER('JOHN.DOE');
END IF;
```

8

APEX_DEBUG

The APEX_DEBUG package provides utility functions for managing the debug message log. Specifically, this package provides the necessary APIs to instrument and debug PL/SQL code contained within your Application Express application as well as PL/SQL code in database stored procedures and functions. Instrumenting your PL/SQL code makes it much easier to track down bugs and isolate unexpected behavior more quickly.

The package also provides the means to enable and disable debugging at different debug levels and utility procedures to clean up the message log.

You can view the message log either as described in the " Accessing Debugging Mode" section of the *Oracle Application Express App Builder User's Guide* or by querying the APEX_DEBUG_MESSAGES view.

For further information, see the individual API descriptions.

Note:

In Oracle Application Express 4.2, the APEX_DEBUG_MESSAGE package was renamed to APEX_DEBUG. The APEX_DEBUG_MESSAGE package name is still supported to provide backward compatibility. As a best practice, however, use the new APEX_DEBUG package for new applications unless you plan to run them in an earlier version of Oracle Application Express.

See Also:

"Accessing Debugging Mode" in *Oracle Application Express App Builder User's Guide*

8.1 Constants

The following constants are used by this package.

```
subtype t_log_level is pls_integer;
c_log_level_error constant t_log_level := 1; -- critical error
c_log_level_warn constant t_log_level := 2; -- less critical error
c_log_level_info constant t_log_level := 4; -- default level if debugging is enabled
(for example, used by apex_application.debug)
c_log_level_app_enter constant t_log_level := 5; -- application: messages when
procedures/functions are entered
c_log_level_app_trace constant t_log_level := 6; -- application: other messages
within procedures/functions
c_log_level_engine_enter constant t_log_level := 8; -- Application Express engine:
messages when procedures/functions are entered
c_log_level_engine_trace constant t_log_level := 9; -- Application Express engine:
other messages within procedures/functions
```

8.2 DISABLE Procedure

This procedure turns off debug messaging.

Syntax

```
APEX_DEBUG.DISABLE;
```

Parameters

None.

Example

This example shows how you can turn off debug messaging.

```
BEGIN
    APEX_DEBUG.DISABLE();
END;
```

See Also:

"ENABLE Procedure (page 8-3)"

8.3 DISABLE_DBMS_OUTPUT Procedure

This procedure stops writing all debug logs also via `dbms_output`.

Syntax

```
disable_dbms_output;
```

Parameters

None.

Example

See `enable_dbms_output`.

See Also:

- "ENABLE_DBMS_OUTPUT Procedure (page 8-5)"
- "ENABLE Procedure (page 8-3)"
- "DISABLE Procedure (page 8-2)"

8.4 ENABLE Procedure

This procedure turns on debug messaging. You can specify, by level of importance, the types of debug messages that are monitored.

Note:

You only need to call `ENABLE` procedure once per page view or page accept.

Syntax

```
APEX_DEBUG.ENABLE (
    p_level    IN  T_LOG_LEVEL DEFAULT C_LOG_LEVEL_INFO );
```

Parameters

Table 8-1 ENABLE Procedure Parameters

Parameter	Description
p_level	Level or levels of messages to log. Must be an integer from 1 to 9, where level 1 is the most important messages and level 4 (the default) is the least important. Setting to a specific level logs messages both at that level and below that level. For example, setting p_level to 2 logs any message at level 1 and 2.

Example

This examples shows how to enable logging of messages for levels 1, 2 and 4. Messages at higher levels are not logged.

```
BEGIN
    APEX_DEBUG.ENABLE(
        apex_debug.c_log_level_info);
END;
```

8.5 ENTER Procedure

This procedure logs messages at level c_log_level_app_enter. Use APEX_DEBUG.ENTER() to log the routine name and it's arguments at the beginning of a procedure or function.

Syntax

```
APEX_DEBUG.ENTER (
p_routine_name IN VARCHAR2,
p_name01 IN VARCHAR2 DEFAULT NULL,
p_value01 IN VARCHAR2 DEFAULT NULL,
p_name02 IN VARCHAR2 DEFAULT NULL,
p_value02 IN VARCHAR2 DEFAULT NULL,
p_name03 IN VARCHAR2 DEFAULT NULL,
p_value03 IN VARCHAR2 DEFAULT NULL,
p_name04 IN VARCHAR2 DEFAULT NULL,
p_value04 IN VARCHAR2 DEFAULT NULL,
p_name05 IN VARCHAR2 DEFAULT NULL,
p_value05 IN VARCHAR2 DEFAULT NULL,
p_name06 IN VARCHAR2 DEFAULT NULL,
p_value06 IN VARCHAR2 DEFAULT NULL,
p_name07 IN VARCHAR2 DEFAULT NULL,
p_value07 IN VARCHAR2 DEFAULT NULL,
p_name08 IN VARCHAR2 DEFAULT NULL,
p_value08 IN VARCHAR2 DEFAULT NULL,
p_name09 IN VARCHAR2 DEFAULT NULL,
p_value09 IN VARCHAR2 DEFAULT NULL,
p_name10 IN VARCHAR2 DEFAULT NULL,
p_value10 IN VARCHAR2 DEFAULT NULL,
p_value_max_length IN PLS_INTEGER DEFAULT 1000 );
```

Parameters

Table 8-2 APEX_DEBUG.ENTER Procedure Parameters

Parameter	Description
p_routine_name	The name of the procedure or function.
p_namexx/p_valuexx	The procedure or function parameter name and value.
p_value_max_length	The p_valuexx values is truncated to this length.

Example

This example shows how to use APEX_ENTER to add a debug message at the beginning of a procedure.

```
procedure foo (
    p_widget_id in number,
    p_additional_data in varchar2,
    p_emp_rec in emp%rowtype )
is
begin
    apex_debug.enter('foo',
        'p_widget_id' , p_widget_id,
        'p_additional_data', p_additional_data,
        'p_emp_rec.id' , p_emp_rec.id );
....do something....
end foo;
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"

8.6 ENABLE_DBMS_OUTPUT Procedure

This procedure writes all debug logs via `dbms_output`. If debug is disabled, this call also enables it with log level `c_log_level_warn`. You have to set a debug level higher than `c_log_level_warn` for finer grained debug output. The output `95` starts with a configurable prefix, followed by the log level, `"|"` and the actual debug message.

Syntax

```
enable_dbms_output (
    p_prefix in varchar2 default '# APEX|' );
```

Parameters

Table 8-3 ENABLE_DBMS_OUTPUT Procedure Parameters

Parameter	Description
p_prefix	Prefix for lines that go to dbms_output, default '# APEX|'.

Example

This sqlplus code writes the debug messages for 4, 5, 7, and 8 via dbms_output.

```
set serveroutput on size unlimited
begin
 apex_debug.error('1');
 apex_debug.warn('2');
 apex_debug.enable_dbms_output(p_prefix=>'Debug-');
 apex_debug.error('4');
 apex_debug.warn('5');
 apex_debug.info('6');
 apex_debug.enable(p_level=>apex_debug.c_log_level_info);
 apex_debug.info('7');
 apex_debug.enable_dbms_output;
 apex_debug.info('8');
 apex_debug.disable_dbms_output;
 apex_debug.info('9');
end;
 /
Output:
  Debug-ERR|4
  Debug-WRN|5
  Debug-INF|7
  # APEX|INF|8
```

See Also:

- "DISABLE_DBMS_OUTPUT Procedure (page 8-3)"
- "ENABLE Procedure (page 8-3)"
- "DISABLE Procedure (page 8-2)"

8.7 ERROR Procedure

This procedure logs messages at level c_log_level_error. This procedure always logs, even if debug mode is turned off.

Syntax

```
APEX_DEBUG.ERROR (
    p_message IN VARCHAR2,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    p3 IN VARCHAR2 DEFAULT NULL,
    p4 IN VARCHAR2 DEFAULT NULL,
```

```
    p5 IN VARCHAR2 DEFAULT NULL,
    p6 IN VARCHAR2 DEFAULT NULL,
    p7 IN VARCHAR2 DEFAULT NULL,
    p8 IN VARCHAR2 DEFAULT NULL,
    p9 IN VARCHAR2 DEFAULT NULL,
    p_max_length IN PLS_INTEGER DEFAULT 1000 );
```

Parameters

Table 8-4 APEX_DEBUG.ERROR Procedure Parameters

Parameter	Description
`p_message`	The debug message. Occurrences of '%s' are replaced by `p0` to `p19`, as in `utl_lms.format_message` and C's sprintf. Occurrences of '%%' represent the special character '%'. Occurrences of '%<n>' are replaced by p<n>.
`p0` through `p9`	Substitution strings for '%s' placeholders.
`p_max_length`	The `p<n>` values are truncated to this length.

Example

This example shows how to use `APEX_ERROR` to log a critical error in the debug log.

```
apex_debug.error('Critical error %s', sqlerrm);
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"

8.8 INFO Procedure

This procedure logs messages at level `c_log_level_info`.

Syntax

```
APEX_DEBUG.INFO (
    p_message IN VARCHAR2,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    p3 IN VARCHAR2 DEFAULT NULL,
    p4 IN VARCHAR2 DEFAULT NULL,
    p5 IN VARCHAR2 DEFAULT NULL,
    p6 IN VARCHAR2 DEFAULT NULL,
    p7 IN VARCHAR2 DEFAULT NULL,
    p8 IN VARCHAR2 DEFAULT NULL,
```

```
p9 IN VARCHAR2 DEFAULT NULL,
p_max_length IN PLS_INTEGER DEFAULT 1000 );
```

Parameters

Table 8-5 APEX_DEBUG.INFO Procedure Parameters

Parameter	Description
`p_message`	The debug message. Occurrences of '%s' are replaced by `p0` to `p19`, as in `utl_lms.format_message` and C's sprintf. Occurrences of '%%' represent the special character '%'. Occurrences of '%<n>' are replaced by p<n>.
`p0` through `p9`	Substitution strings for '%s' placeholders.
`p_max_length`	The `p<n>` values are truncated to this length.

Example

This example shows how to use `APEX_DEBUG.INFO` to log information in the debug log.

```
apex_debug.info('Important: %s', 'fnord');
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "ENTER Procedure (page 8-4)"

8.9 LOG_DBMS_OUTPUT Procedure

This procedure writes the contents of `dbms_output.get_lines` to the debug log. Messages of legacy applications which use `dbms_output` are copied into the debug log. In order to write to the debug log, `dbms_output.enable` must be performed

Syntax

```
APEX_DEBUG.LOG_DBMS_OUTPUT;
```

Parameters

None.

Example

This example shows how to log the contents of the `DBMS_OUTPUT` buffer in the debug log.

```
sys.dbms_output.enable;
sys.dbms_output.put_line('some data');
sys.dbms_output.put_line('other data');
apex_debug.log_dbms_output;
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"

8.10 LOG_LONG_MESSAGE Procedure

Use this procedure to emit debug messages from PLSQL components of Application Express, or PLSQL procedures and functions. This procedure is the same as LOG_MESSAGE, except it allows logging of much longer messages, which are subsequently split into 4,000 character chunks in the debugging output (because a single debug message is constrained to 4,000 characters).

Note:

Instead of this procedure, use "ERROR Procedure (page 8-6)," "WARN Procedure (page 8-17)," "MESSAGE Procedure (page 8-12)," "INFO Procedure (page 8-7)," "ENTER Procedure (page 8-4)," or "TRACE Procedure (page 8-16)."

Syntax

```
APEX_DEBUG.LOG_LONG_MESSAGE (
    p_message    IN VARCHAR2  DEFAULT NULL,
    p_enabled    IN BOOLEAN   DEFAULT FALSE,
    p_level      IN T_LOG_LEVEL DEFAULT C_LOG_LEVEL_APP_TRACE);
```

Parameters

Table 8-6 APEX_DEBUG.LOG_LONG_MESSAGE Procedure Parameters

Parameter	Description
p_message	Log long message with maximum size of 32767 bytes.
p_enabled	Set to TRUE to always log messages, irrespective of whether debugging is enabled. Set to FALSE to only log messages if debugging is enabled.
p_level	Identifies the level of the long log message. See "Constants (page 8-2)."

Example

This example shows how to enable debug message logging for 1 and 2 level messages and display a level 1 message that could contain anything up to 32767 characters. Note, the `p_enabled` parameter need not be specified, as debugging has been explicitly enabled and the default of FALSE for this parameter respects this enabling.

```
DECLARE
    l_msg VARCHAR2(32767) := 'Debug outputs anything up to varchar2 limit';
BEGIN
    APEX_DEBUG.ENABLE (p_level => 2);

    APEX_DEBUG.LOG_LONG_MESSAGE(
        p_message => l_msg,
        p_level => 1 );
END;
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"

8.11 LOG_MESSAGE Procedure [Deprecated]

This procedure logs a debug message.

Note:

Instead of this procedure, use "ERROR Procedure (page 8-6)," "WARN Procedure (page 8-17)," "MESSAGE Procedure (page 8-12)," "INFO Procedure (page 8-7)," "ENTER Procedure (page 8-4)," or "TRACE Procedure (page 8-16)."

Syntax

```
APEX_DEBUG.LOG_MESSAGE (
    p_message IN VARCHAR2 DEFAULT NULL,
    p_enabled IN BOOLEAN DEFAULT FALSE,
    p_level IN T_LOG_LEVEL DEFAULT C_LOG_LEVEL_APP_TRACE );
```

Parameters

Table 8-7 APEX_DEBUG.LOG_MESSAGE Procedure Parameters

Parameter	Description
`p_message`	The debug message with a maximum length of 1000 bytes.

Table 8-7 (Cont.) APEX_DEBUG.LOG_MESSAGE Procedure Parameters

Parameter	Description
p_enabled	Messages are logged when logging is enabled, setting a value of TRUE enables logging.
p_level	Identifies the level of the log message where 1 is most important and 9 is least important. This is an integer value.

Example

This example shows how to enable debug message logging for 1 and 2 level messages and display a level 1 message showing a variable value. Note, the p_enabled parameter need not be specified, as debugging has been explicitly enabled and the default of FALSE for this parameter respects this enabling.

```
DECLARE
    l_value varchar2(100) := 'test value';
BEGIN
    APEX_DEBUG.ENABLE (p_level => 2);

APEX_DEBUG.LOG_MESSAGE(
    p_message => 'l_value = ' || l_value,
    p_level => 1 );

END;
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"

8.12 LOG_PAGE_SESSION_STATE Procedure

This procedure logs the session's item values.

Syntax

```
APEX_DEBUG.LOG_PAGE_SESSION_STATE (
    p_page_id IN NUMBER DEFAULT NULL,
    p_enabled IN BOOLEAN DEFAULT FALSE,
    p_level IN T_LOG_LEVEL DEFAULT C_LOG_LEVEL_APP_TRACE );
```

Parameters

Table 8-8 APEX_DEBUG.LOG_SESSION_STATE Procedure Parameters

Parameter	Description
p_page_id	Identifies a page within the current applicaton and workspace context.
p_enabled	Messages are logged when logging is enabled, setting a value of TRUE enables logging.
p_level	Identifies the level of the log message where 1 is most important, 9 is least important. Must be an integer value.

Example

This example shows how to enable debug message logging for 1 and 2 level messages and display a level 1 message containing all the session state for the application's current page. Note, the p_enabled parameter need not be specified, as debugging has been explicitly enabled and the default of FALSE for this parameter respects this enabling. Also note the p_page_id has not been specified, as this example just shows session state information for the application's current page.

```
BEGIN
    APEX_DEBUG.ENABLE (p_level => 2);

    APEX_DEBUG.LOG_PAGE_SESSION_STATE (p_level => 1);

END;
```

8.13 MESSAGE Procedure

This procedure logs a formatted debug message, general version.

Syntax

```
APEX_DEBUG.MESSAGE (
    p_message IN VARCHAR2,
    p0  IN VARCHAR2 DEFAULT NULL,
    p1  IN VARCHAR2 DEFAULT NULL,
    p2  IN VARCHAR2 DEFAULT NULL,
    p3  IN VARCHAR2 DEFAULT NULL,
    p4  IN VARCHAR2 DEFAULT NULL,
    p5  IN VARCHAR2 DEFAULT NULL,
    p6  IN VARCHAR2 DEFAULT NULL,
    p7  IN VARCHAR2 DEFAULT NULL,
    p8  IN VARCHAR2 DEFAULT NULL,
    p9  IN VARCHAR2 DEFAULT NULL,
    p10 IN VARCHAR2 DEFAULT NULL,
    p11 IN VARCHAR2 DEFAULT NULL,
    p12 IN VARCHAR2 DEFAULT NULL,
    p13 IN VARCHAR2 DEFAULT NULL,
    p14 IN VARCHAR2 DEFAULT NULL,
    p15 IN VARCHAR2 DEFAULT NULL,
    p16 IN VARCHAR2 DEFAULT NULL,
    p17 IN VARCHAR2 DEFAULT NULL,
    p18 IN VARCHAR2 DEFAULT NULL,
    p19 IN VARCHAR2 DEFAULT NULL,
```

```
    p_max_length IN PLS_INTEGER DEFAULT 1000,
    p_level IN T_LOG_LEVEL DEFAULT C_LOG_LEVEL_INFO,
    p_force IN BOOLEAN DEFAULT FALSE );
```

Parameters

Table 8-9 APEX_DEBUG.MESSAGE Procedure Parameters

Parameter	Description
`p_message`	The debug message. Occurrences of '%s' is replaced by `p0` to `p19`, as in `utl_lms.format_message` and C's sprintf. Occurrences of '%%' represent the special character '%'. Occurrences of '%<n>' are replaced by `p<n>`.
`p0` through `p19`	Substitution strings for '%s' placeholders.
`p_max_length`	The `p<n>` values is truncated to this length.
`p_level`	The log level for the message, default is `c_log_level_info`. See "Constants (page 8-2)".
`p_force`	If TRUE, this generates a debug message even if the page is not rendered in debug mode or `p_level` is greater than the configured debug messaging (using the URL or using the enable procedure).

Example

This example shows how to use the `APEX_DEBUG.MESSAGE` procedure to add text to the debug log.

```
apex_debug.message('the value of %s + %s equals %s', 3, 5, 'eight');
```

See Also:

- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"
- "ENTER Procedure (page 8-4)"

8.14 REMOVE_DEBUG_BY_AGE Procedure

Use this procedure to delete from the debug message log all data older than the specified number of days.

Syntax

```
APEX_DEBUG.REMOVE_DEBUG_BY_AGE (
    p_application_id IN NUMBER,
    p_older_than_days IN NUMBER);
```

Parameters

Table 8-10 APEX_DEBUG.REMOVE_DEBUG_BY_AGE Procedure Parameters

Parameter	Description
p_application_id	The application ID of the application.
p_older_than_days	The number of days data can exist in the debug message log before it is deleted.

Example

This example demonstrates removing debug messages relating to the current application, that are older than 3 days old.

```
BEGIN
    APEX_DEBUG.REMOVE_DEBUG_BY_AGE (
        p_application_id  => TO_NUMBER(:APP_ID),
        p_older_than_days => 3 );
END;
```

8.15 REMOVE_DEBUG_BY_APP Procedure

Use this procedure to delete from the debug message log all data belonging to a specified application.

Syntax

```
APEX_DEBUG.REMOVE_DEBUG_BY_APP (
    p_application_id IN NUMBER);
```

Parameters

Table 8-11 APEX_DEBUG.REMOVE_DEBUG_BY_APP Procedure Parameters

Parameter	Description
p_application_id	The application ID of the application.

Example

This example demonstrates removing all debug messages logged for the current application.

```
BEGIN
    APEX_DEBUG.REMOVE_DEBUG_BY_APP(
        p_application_id => TO_NUMBER(:APP_ID) );
END;
```

8.16 REMOVE_DEBUG_BY_VIEW Procedure

Use this procedure to delete all data for a specified view from the message log.

Syntax

```
APEX_DEBUG.REMOVE_DEBUG_BY_VIEW (
    p_application_id IN NUMBER,
    p_view_id IN NUMBER);
```

Parameters

Table 8-12 APEX_DEBUG.REMOVE_DEBUG_BY_VIEW Procedure Parameters

Parameter	Description
p_application_id	The application ID of the application.
p_view_id	The view ID of the view.

Example

This example demonstrates the removal of debug messages within the 'View Identifier' of 12345, belonging to the current application.

```
BEGIN
    APEX_DEBUG.REMOVE_DEBUG_BY_VIEW (
        p_application_id => TO_NUMBER(:APP_ID),
        p_view_id        => 12345 );
END;
```

8.17 REMOVE_SESSION_MESSAGES Procedure

This procedure deletes from the debug message log all data for a given session in your workspace defaults to your current session.

Syntax

```
APEX_DEBUG.REMOVE_SESSION_MESSAGES (
    p_session    IN NUMBER  DEFAULT NULL);
```

Parameters

Table 8-13 APEX_DEBUG.REMOVE_SESSION_MESSAGES Procedure Parameters

Parameter	Description
p_session	The session ID. Defaults to your current session.

Example

This example demonstrates the removal of all debug messages logged within the current session. Note: As no value is passed for the p_session parameter, the procedure defaults to the current session.

```
BEGIN
    APEX_DEBUG.REMOVE_SESSION_MESSAGES();
END;
```

8.18 TOCHAR Function

This procedure converts a BOOLEAN to a VARCHAR2.

Syntax

```
APEX_DEBUG.TOCHAR (
    p_value IN BOOLEAN )
    return VARCHAR2;
```

Parameters

Table 8-14 APEX_DEBUG.TOCHAR Function Parameters

Parameter	Description
p_value	A BOOLEAN 0or 1 that is converted to FALSE or TRUE respectively.

Example

This example shows how to use the APEX_DEBUG.TOCHAR function to convert boolean values to varchar2, so they can be passed to the other debug procedures.

```
declare
    l_state boolean;
begin
    ....
    apex_debug.info('Value of l_state is %s', apex_debug.tochar(l_state));
    ....
end;
```

8.19 TRACE Procedure

This procedure logs messages at level c_log_level_app_trace.

Syntax

```
APEX_DEBUG.TRACE (
    p_message IN VARCHAR2,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    p3 IN VARCHAR2 DEFAULT NULL,
    p4 IN VARCHAR2 DEFAULT NULL,
    p5 IN VARCHAR2 DEFAULT NULL,
    p6 IN VARCHAR2 DEFAULT NULL,
    p7 IN VARCHAR2 DEFAULT NULL,
    p8 IN VARCHAR2 DEFAULT NULL,
    p9 IN VARCHAR2 DEFAULT NULL,
    p_max_length IN PLS_INTEGER DEFAULT 1000 );
```

Parameters

Table 8-15 APEX_DEBUG.TRACE Procedure Parameters

Parameter	Description
p_message	The debug message. Occurrences of '%s' are replaced by p0 to p19, as in utl_lms.format_message and C's sprintf. Occurrences of '%%' represent the special character '%'. Occurrences of '%<n>' are replaced by p<n>.
p0 through p9	Substitution strings for '%s' placeholders.
p_max_length	The p<n> values are truncated to this length.

Example

This example shows how to use APEX_DEBUG.TRACE to log low-level debug information in the debug log.

```
apex_debug.trace('Low-level information: %s+%s=%s', 1, 2, 3);
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "WARN Procedure (page 8-17)"
- "ENTER Procedure (page 8-4)"
- "INFO Procedure (page 8-7)"

8.20 WARN Procedure

This procedure logs messages at level c_log_level_warn.

Syntax

```
APEX_DEBUG.WARN (
    p_message IN VARCHAR2,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    p3 IN VARCHAR2 DEFAULT NULL,
    p4 IN VARCHAR2 DEFAULT NULL,
    p5 IN VARCHAR2 DEFAULT NULL,
    p6 IN VARCHAR2 DEFAULT NULL,
    p7 IN VARCHAR2 DEFAULT NULL,
    p8 IN VARCHAR2 DEFAULT NULL,
    p9 IN VARCHAR2 DEFAULT NULL,
    p_max_length IN PLS_INTEGER DEFAULT 1000 );
```

Parameters

Table 8-16 APEX_DEBUG.WARN Procedure Parameters

Parameter	Description
`p_message`	The debug message. Occurrences of '%s' are replaced by `p0` to `p19`, as in `utl_lms.format_message` and C's sprintf. Occurrences of '%%' represent the special character '%'. Occurrences of '%<n>' are replaced by p<n>.
`p0` through `p9`	Substitution strings for '%s' placeholders.
`p_max_length`	The `p<n>` values are truncated to this length.

Example

This example shows how to use `APEX_DEBUG.WARN` to log highly important data in the debug log.

```
apex_debug.warn('Soft constraint %s violated: %s', 4711, sqlerrm);
```

See Also:

- "MESSAGE Procedure (page 8-12)"
- "ERROR Procedure (page 8-6)"
- "ENTER Procedure (page 8-4)"
- "TRACE Procedure (page 8-16)"
- "INFO Procedure (page 8-7)"

9

APEX_ESCAPE

The APEX_ESCAPE package provides functions for escaping special characters in strings to ensure that the data is suitable for further processing.

9.1 Constants

The APEX_ESCAPE package uses the following constants.

```
c_ldap_dn_reserved_chars constant varchar2(8) := '"+,;<=>\';
c_ldap_search_reserved_chars constant varchar2(5) := '*()\/';
c_html_whitelist_tags constant varchar2(255) := '<h1>,</h1>,<h2>,</h2>,<h3>,</
h3>,<h4>,</h4>,<p>,</p>,<b>,</b>,<strong>,</strong>,<i>,</i>,<ul>,</ul>,<ol>,</
ol>,<li>,</li>,<br />,<hr/>';
```

9.2 HTML Function

This function escapes characters which can change the context in an html environment. It is an extended version of the well-known `sys.htf.escape_sc`.

The function's result depends on the escaping mode that is defined by using `apex_escape.set_html_escaping_mode`. By default, the escaping mode is "`Extended`", but it can be overridden by manually calling `set_html_escaping_mode` or by setting the application security attribute "`HTML Escaping Mode`" to "`Basic`". If the mode is "`Basic`", the function behaves like `sys.htf.escape_sc`. Otherwise, the rules below apply.

The following table, Table 9-1 (page 9-2), depicts ascii characters that the function transforms and their escaped values:

Table 9-1 Escaped Values for Transformed ASCII Characters

Raw ASCI Characters	Returned Escaped Characters
&	&amp;
"	&quot;
<	&lt;
>	&gt;
'	&#x27;
/	&#x2F;

Syntax

```
APEX_ESCAPE.HTML (
    p_string IN VARCHAR2 )
    return VARCHAR2;
```

Parameters

Table 9-2 (page 9-2) describes the parameters available in the HTML function.

Table 9-2 HTML Function Parameters

Parameter	Description
p_string	The string text that is escaped

Example

This example tests escaping in basic ('B') and extended ('E') mode.

```
declare
procedure eq(p_str1 in varchar2,p_str2 in varchar2)
    is
    begin
        if p_str1||'.' <> p_str2||'.' then
            raise_application_error(-20001,p_str1||' <> '||p_str2);
    end if;
end eq;
begin
    apex_escape.set_html_escaping_mode('B');
    eq(apex_escape.html('hello &"<>''/'), 'hello &amp;&quot;&lt;&gt;''/');
    apex_escape.set_html_escaping_mode('E');
    eq(apex_escape.html('hello &"<>''/'), 'hello
    &amp;&quot;&lt;&gt;&#x27;&#x2F;');
end;
```

See Also:

- "HTML_TRUNC Function (page 9-3)"
- "HTML_WHITELIST Function (page 9-4)"
- "HTML_ATTRIBUTE Function (page 9-3)"
- "SET_HTML_ESCAPING_MODE Procedure (page 9-10)"

9.3 HTML_ATTRIBUTE Function

Use this function to escape the values of html entity attributes. It hex escapes everything that is not alphanumeric or in one of the following characters ',' '.' '-' ' ' '_' .

Syntax

```
APEX_ESCAPE.HTML_ATTRIBUTE (
    p_string IN VARCHAR2 )
    return VARCHAR2;
```

Parameters

Table 9-3 HTML_ATTRIBUTE Function Parameters

Parameter	Description
p_string	The text string that is escaped.

Example

See " HTML_TRUNC Function".

See Also:

- "HTML_TRUNC Function (page 9-3)"
- "HTML Function (page 9-1)"
- "HTML_WHITELIST Function (page 9-4)"
- "SET_HTML_ESCAPING_MODE Procedure (page 9-10)"

9.4 HTML_TRUNC Function

The HTML_TRUNC function escapes html and limits the returned string to p_length bytes. This function returns the first p_length bytes of an input clob and escapes them. You can use this function if the input clob might be too large to fit in a varchar2 variable and it is sufficient to only display the first part of it.

Syntax

```
APEX_ESCAPE.HTML_TRUNC (
    p_string IN CLOB,
```

```
    p_length IN NUMBER DEFAULT 4000 )
    return VARCHAR2;
```

Parameters

Table 9-4 HTML_TRUNC Function Parameters

Parameter	Description
p_string	The text string that is escaped.
p_length	The number of bytes from p_string that are escaped.

Example

This example generates a html list of of titles and text bodies. Html entity attributes are escaped with HTML_ATTRIBUTE, whereas normal text is escaped with HTML and HTML_TRUNC.

```
begin
    htp.p('<ul>');
    for l_data in ( select title, cls, body
        from my_topics )
    loop
    sys.htp.p('<li><span class="'||
        apex_escape.html_attribute(l_data.cls)||'">'||
        apex_escape.html(l_data.title)||'</span>');
    sys.htp.p(apex_escape.html_trunc(l_data.body));
    sys.htp.p('</li>');
    end loop;
    htp.p('</ul>');
end;
```

See Also:

- "HTML_ATTRIBUTE Function (page 9-3)"
- "HTML Function (page 9-1)"
- "HTML_WHITELIST Function (page 9-4)"
- "SET_HTML_ESCAPING_MODE Procedure (page 9-10)"

9.5 HTML_WHITELIST Function

The HTML_WHITELIST function performs HTML escape on all characters in the input text except the specified whitelist tags. This function can be useful if the input text contains simple html markup but a developer wants to ensure that an attacker cannot use malicious tags for cross-site scripting.

Syntax

```
APEX_ESCAPE.HTML_WHITELIST (
    p_html IN VARCHAR2,
    p_whitelist_tags IN VARCHAR2 DEFAULT c_html_whitelist_tags )
    return VARCHAR2;
```

Parameters

Table 9-5 HTML_WHITELIST Function Parameters

Parameter	Description
p_html	The text string that is filtered.
p_whitelist_tags	The comma separated list of tags that stays in p_html.

Example

This example shows how to use HTML_WHITELIST to remove unwanted html markup from a string, while preserving whitelisted tags.

```
begin
sys.htp.p(apex_escape.html_whitelist(         '<h1>Hello<script>alert("XSS");</
script></h1>')); end;
```

See Also:

- "HTML_ATTRIBUTE Function (page 9-3)"
- "HTML Function (page 9-1)"
- "HTML_TRUNC Function (page 9-3)"
- "SET_HTML_ESCAPING_MODE Procedure (page 9-10)"

9.6 JS_LITERAL Function

The JS_LITERAL function escapes and optionally enquotes a javascript string. This function replaces non-immune characters with \xHH or \uHHHH equivalents. The result can be injected into javascript code, within <script> tags or inline ("javascript:xxx"). Immune characters include a through z, A through Z, 0 through 9, commas ",", periods "." and underscores "_" if the output should not be enclosed in quotes when the parameter p_quote is null. If p_quote has a value, printable ASCII 7 characters except for & < > " ' ` / \ % are not escaped.

Syntax

```
APEX_ESCAPE.JS_LITERAL (
    p_string IN VARCHAR2,
    p_quote  IN VARCHAR2 DEFAULT '''' )
    return VARCHAR2;
```

Parameters

Table 9-6 JS_LITERAL Function Parameters

Parameter	Description
p_string	The text string that is escaped.

Table 9-6 (Cont.) JS_LITERAL Function Parameters

Parameter	Description
p_quote	If not null, this string is placed on the left and right of the result. The quotation character must be a single or a double quotation mark.

Example

It describes how to use JS_LITERAL to escape special characters in the l_string variable.

```
declare
    l_string varchar2(4000) := 'O''Brien';
begin
    sys.htp.p('<script>'||
        'alert('||apex_escape.js_literal(l_string)||');'||'</script>');
end;
```

9.7 JSON Function

This function returns p_string with all special characters escaped.

Syntax

```
APEX_ESCAPE.JSON (
    p_string  IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Table 9-7 JSON Function Parameters

Parameter	Description
p_string	The string to be escaped.

Returns/Raised Errors

Table 9-8 JSON Function Returns

Return	Description
VARCHAR2	The escaped string.

Example

The following example prints this: { "name": "O\u0027Brien"}

```
declare
    l_string varchar2(4000) := 'O''Brien';
begin
    sys.htp.p('{ "name": "'||apex_escape.json(l_string)||'"}');
end;
```

9.8 LDAP_DN Function

The LDAP_DN function escapes reserved characters in an LDAP distinguished name, according to RFC 4514. The RFC describes "+,;<=>\ as reserved characters (see p_reserved_chars). These are escaped by a backslash, for example, " becomes \". Non-printable characters, ascii 0 - 31, and ones with a code > 127 (see p_escape_non_ascii) are escaped as \xx, where xx is the hexadecimal character code. The space character at the beginning or end of the string and a # at the beginning is also escaped with a backslash.

Syntax

```
APEX_ESCAPE.LDAP_DN (
    p_string IN VARCHAR2,
    p_reserved_chars IN VARCHAR2 DEFAULT c_ldap_dn_reserved_chars,
    p_escaped_non_ascii IN BOOLEAN DEFAULT TRUE )
    return VARCHAR2;
```

Parameters

Table 9-9 LDAP_DN Function Parameters

Parameter	Description
p_string	The text string that is escaped.
p_reserved_chars	A list of characters that when found in p_string is escaped with a backslash.
p_escaped_non_ascii	If TRUE, characters above ascii 127 in p_string are escaped with a backslash. This is supported by RFCs 4514 and 2253, but may cause errors with older LDAP servers and Microsoft AD.

Example

This example escapes characters in l_name and places the result in l_escaped.

```
declare
    l_name varchar2(4000) := 'Joe+User';
    l_escaped varchar2(4000);
begin
    l_escaped := apex_escape.ldap_dn(l_name);
    htp.p(l_name||' becomes '||l_escaped);
end;
```

Note:

"LDAP_SEARCH_FILTER Function (page 9-7)"

9.9 LDAP_SEARCH_FILTER Function

The LDAP_SEARCH_FILTER function escapes reserved characters in an LDAP search filter, according to RFC 4515. The RFC describes *()\/ as reserved characters (see p_reserved_chars). These, non-printable characters (ascii 0 - 31) and ones with a

code > 127 (see p_escape_non_ascii) are escaped as \xx, where xx is the hexadecimal character code.

Syntax

```
APEX_ESCAPE.LDAP_SEARCH_FILTER (
    p_string              IN VARCHAR2,
    p_reserved_chars    IN VARCHAR2 DEFAULT c_ldap_search_reserved_chars,
    p_escape_non_ascii IN BOOLEAN DEFAULT TRUE )
    return VARCHAR2;
```

Parameters

Table 9-10 LDAP_SEARCH_FILTER Function Parameters

Parameter	Description
p_string	The text string that is escaped.
p_reserved_chars	A list of characters that when found in p_string is escaped with \xx where xx is the character's ASCII hexadecimal code.
p_escape_non_ascii	If TRUE, characters above ascii 127 in p_string are escaped with \xx where xx is the character's ASCII hexadecimal code. This is supported by RFCs 4514, but may cause errors with older LDAP servers and Microsoft AD.

Example

This example escapes the text in l_name and places the result in l_escaped.

```
declare
l_name varchar2(4000) := 'Joe*User';
l_escaped varchar2(4000);
begin
    l_escaped := apex_escape.ldap_search_filter(l_name);
    htp.p(l_name||' becomes '||l_escaped);
end;
```

Note:

"LDAP_DN Function (page 9-7)"

9.10 NOOP Function

Return p_string unchanged. Use this function to silence automatic injection detection tests, similar to dbms_assert.noop for SQL injection.

Syntax

```
APEX_ESCAPE.NOOP (
    p_string IN VARCHAR2)
    return VARCHAR2 deterministic;
```

Parameters

Table 9-11 APEX_ESCAPE.NOOP Function Parameters

Parameter	Description
p_string	The input text string.

Example

This example shows how to use NOOP to show the developer's intention to explicitly not escape text.

```
begin
    sys.htp.p(apex_escape.noop('Cats & Dogs'));
end;
```

9.11 REGEXP Function

This function escapes characters that can change the context in a regular expression. It should be used to secure user input. The following list depicts ascii characters that the function escapes with a backslash (\):

\.^$*+-?()[]{|

Syntax

```
APEX_ESCAPE.REGEXP (
    p_string IN VARCHAR2);
```

Parameters

Table 9-12 APEX_ESCAPE.REGEXP Function Parameters

Parameter	Description
p_string	Text to escape.

Example

The following example ensures the special character "-" in Mary-Ann will be escaped and ignored by the regular expression engine.

```
declare
    l_subscribers varchar2(4000) := 'Christina,Hilary,Mary-Ann,Joel';
    l_name varchar2(4000) := 'Mary-Ann';
begin
    if regexp_instr(l_subscribers,'(^|,)'|| apex_escape.regexp(l_name)||'($|,)')>0
    then
        sys.htp.p('found');
    else
        sys.htp.p('not found')
    endif;
end
```

9.12 SET_HTML_ESCAPING_MODE Procedure

The `SET_HTML_ESCAPING_MODE` procedure configures HTML escaping mode for `apex_escape.html`.

Syntax

```
APEX_ESCAPE.SET_HTML_ESCAPING_MODE (
    p_mode IN VARCHAR2);
```

Parameters

Table 9-13 APEX_ESCAPE.SET_HTML_ESCAPING_MODE Procedure Parameters

Parameter	Description
`p_mode`	If equal to `B`, then do basic escaping, like `sys.htf.escape_sc`. If equal to `E`, then do extended escaping.

Example

For an example, see " HTML Function".

See Also:

- "HTML_WHITELIST Function (page 9-4)"
- "HTML Function (page 9-1)"
- "HTML_TRUNC Function (page 9-3)"
- "HTML_ATTRIBUTE Function (page 9-3)"

10

APEX_ERROR

The APEX_ERROR package provides the interface declarations and some utility functions for an error handling function and includes procedures and functions to raise errors in an Application Express application.

10.1 Constants and Attributes used for Result Types

The following constants are used for the API parameter `p_display_location` and the attribute `display_location` in the `t_error` and `t_error_result` types.

```
c_inline_with_field           constant varchar2(40):='INLINE_WITH_FIELD';
c_inline_with_field_and_notif  constant
varchar2(40):='INLINE_WITH_FIELD_AND_NOTIFICATION';
c_inline_in_notification      constant varchar2(40):='INLINE_IN_NOTIFICATION';
c_on_error_page               constant varchar2(40):='ON_ERROR_PAGE';
```

The following constants are used for the API parameter `associated_type` in the `t_error` type.

```
c_ass_type_page_item          constant varchar2(30):='PAGE_ITEM';
c_ass_type_region             constant varchar2(30):='REGION';
c_ass_type_region_column      constant varchar2(30):='REGION_COLUMN';
```

The following record structure is passed into an error handling callout function and contains all the relevant information about the error.

```
type t_error is record (
    message  varchar2(32767),             /* Error message which will be displayed */
    additional_info varchar2(32767),      /* Only used for display_location
ON_ERROR_PAGE to display additional error information */
    display_location  varchar2(40),       /* Use constants "used for
display_location" below */
    association_type  varchar2(40),       /* Use constants "used for
asociation_type" below */
    page_item_name    varchar2(255),      /* Associated page item name */
    region_id         number,             /* Associated tabular form region id of
the primary application */
    column_alias      varchar2(255),      /* Associated tabular form column alias */
    row_num           pls_integer,        /* Associated tabular form row */
    is_internal_error boolean,            /* Set to TRUE if it's a critical error
raised by the APEX engine, like an invalid SQL/PLSQL statements, ... Internal Errors
are always displayed on the Error Page */
    apex_error_code   varchar2(255),      /* Contains the system message code if
it's an error raised by APEX */
    ora_sqlcode       number,             /* SQLCODE on exception stack which
triggered the error, NULL if the error was not raised by an ORA error */
    ora_sqlerrm      varchar2(32767),     /* SQLERRM which triggered the error, NULL
if the error was not raised by an ORA error */
   error_backtrace   varchar2(32767),     /* Output of
sys.dbms_utility.format_error_backtrace or sys.dbms_utility.format_call_stack */
   error_statement   varchar2(32767),     /* Statement that was parsed when the
error occurred - only suitable when parsing caused the error */
   component         apex.t_component /* Component which has been processed when the
error occurred */
);
```

The following record structure must be returned by an error handling callout function.

```
type t_error_result is record (
   message          varchar2(32767), /* Error message which will be displayed */
   additional_info  varchar2(32767), /* Only used for display_location ON_ERROR_PAGE
to display additional error information */
   display_location varchar2(40),    /* Use constants "used for display_location"
below */
   page_item_name   varchar2(255),   /* Associated page item name */
   column_alias     varchar2(255)    /* Associated tabular form column alias */
);
```

10.2 Example of an Error Handling Function

The following is an example of an error handling function.

```
create or replace function apex_error_handling_example (
    p_error in apex_error.t_error )
    return apex_error.t_error_result
is
    l_result          apex_error.t_error_result;
    l_reference_id    number;
    l_constraint_name varchar2(255);
begin
    l_result := apex_error.init_error_result (
                    p_error => p_error );

    -- If it's an internal error raised by APEX, like an invalid statement or
    -- code which can't be executed, the error text might contain security sensitive
    -- information. To avoid this security problem we can rewrite the error to
    -- a generic error message and log the original error message for further
```

```
    -- investigation by the help desk.
    if p_error.is_internal_error then
        -- mask all errors that are not common runtime errors (Access Denied
        -- errors raised by application / page authorization and all errors
        -- regarding session and session state)
        if not p_error.is_common_runtime_error then
            -- log error for example with an autonomous transaction and return
            -- l_reference_id as reference#
            -- l_reference_id := log_error (
            --                       p_error => p_error );
            --

            -- Change the message to the generic error message which doesn't expose
            -- any sensitive information.
            l_result.message         := 'An unexpected internal application error
has occurred. '||
                                        'Please get in contact with XXX and provide
'||
                                        'reference# '||to_char(l_reference_id,
'999G999G999G990')||
                                        ' for further investigation.';
            l_result.additional_info := null;
        end if;
    else
        -- Always show the error as inline error
        -- Note: If you have created manual tabular forms (using the package
        --       apex_item/htmldb_item in the SQL statement) you should still
        --       use "On error page" on that pages to avoid loosing entered data
        l_result.display_location := case
                                       when l_result.display_location =
apex_error.c_on_error_page then apex_error.c_inline_in_notification
                                       else l_result.display_location
                                     end;

        --
        -- Note: If you want to have friendlier ORA error messages, you can also
define
        --       a text message with the name pattern APEX.ERROR.ORA-number
        --       There is no need to implement custom code for that.
        --

        -- If it's a constraint violation like
        --
        --   -) ORA-00001: unique constraint violated
        --   -) ORA-02091: transaction rolled back (-> can hide a deferred
constraint)
        --   -) ORA-02290: check constraint violated
        --   -) ORA-02291: integrity constraint violated - parent key not found
        --   -) ORA-02292: integrity constraint violated - child record found
        --
        -- we try to get a friendly error message from our constraint lookup
configuration.
        -- If we don't find the constraint in our lookup table we fallback to
        -- the original ORA error message.
        if p_error.ora_sqlcode in (-1, -2091, -2290, -2291, -2292) then
            l_constraint_name := apex_error.extract_constraint_name (
                                     p_error => p_error );

            begin
                select message
                  into l_result.message
```

```
                 from constraint_lookup
                where constraint_name = l_constraint_name;
            exception when no_data_found then null; -- not every constraint has to
be in our lookup table
            end;
        end if;

        -- If an ORA error has been raised, for example a
raise_application_error(-20xxx, '...')
        -- in a table trigger or in a PL/SQL package called by a process and we
        -- haven't found the error in our lookup table, then we just want to see
        -- the actual error text and not the full error stack with all the ORA error
numbers.
        if p_error.ora_sqlcode is not null and l_result.message = p_error.message
then
            l_result.message := apex_error.get_first_ora_error_text (
                                    p_error => p_error );
        end if;

        -- If no associated page item/tabular form column has been set, we can use
        -- apex_error.auto_set_associated_item to automatically guess the affected
        -- error field by examine the ORA error for constraint names or column names.
        if l_result.page_item_name is null and l_result.column_alias is null then
            apex_error.auto_set_associated_item (
                p_error        => p_error,
                p_error_result => l_result );
        end if;
    end if;

    return l_result;
end apex_error_handling_example;
```

10.3 ADD_ERROR Procedure Signature 1

This procedure adds an error message to the error stack that is used to display an error on an error page or inline in a notification. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_message          in varchar2,
    p_additional_info  in varchar2 default null,
    p_display_location in varchar2 );
```

Parameters

Table 10-1 ADD_ERROR Procedure Signature 1 Parameters

Parameters	Description
p_message	Displayed error message.
p_additional_info	Additional error information needed if the error is displayed on the error page.
p_display_location	Specifies where the error message is displayed. Use the constant apex_error.c_inline_notification or apex_error.c_error_page. See "Constants and Attributes used for Result Types (page 10-1)."

Example

This example illustrates how to add a custom error message to the error stack. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_message          => 'This custom account is not active!',
    p_display_location => apex_error.c_inline_in_notification );
```

10.4 ADD_ERROR Procedure Signature 2

This procedure adds an error message to the error stack that is used to display an error for a page item inline in a notification. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of apex_error.c_on_error_page.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_message          in varchar2,
    p_additional_info  in varchar2 default null,
    p_display_location in varchar2,
    p_page_item_name   in varchar2);
```

Parameters

Table 10-2 ADD_ERROR Procedure Signature 2 Parameters

Parameters	Description
p_message	Displayed error message.

Table 10-2 (Cont.) ADD_ERROR Procedure Signature 2 Parameters

Parameters	Description
p_additional_info	Additional error information needed if the error is displayed on the error page.
p_display_location	Specifies where the error message is displayed. Use the constant apex_error.c_inline_with field or apex_error.c_inline_with_field_and_notif. See "Constants and Attributes used for Result Types (page 10-1)."
p_page_item_name	Name of the page item on the current page that is highlighted if apex_error.c_inline_with_field or apex_error.c_inline_with_field_and_notif are used as the display location.

Example

This example illustrates how to add a custom error message to the error stack. The P5_CUSTOMER_ID item is highlighted on the page. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_message          => 'Invalid Customer ID!',
    p_display_location =>  apex_error.c_inline_with_field_and_notif,
    p_page_item_name   => 'P5_CUSTOMER_ID');
```

10.5 ADD_ERROR Procedure Signature 3

This procedure adds an error message to the error stack that is used to display text as defined by a shared component. This error message can be displayed to all display locations. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of apex_error.c_on_error_page.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_error_code          in varchar2,
    p0                    in varchar2 default null,
    p1                    in varchar2 default null,
    p2                    in varchar2 default null,
    p3                    in varchar2 default null,
    p4                    in varchar2 default null,
    p5                    in varchar2 default null,
    p6                    in varchar2 default null,
    p7                    in varchar2 default null,
    p8                    in varchar2 default null,
    p9                    in varchar2 default null,
```

```
    p_escape_placeholders in boolean  default true,
    p_additional_info     in varchar2 default null,
    p_display_location    in varchar2,
    p_page_item_name      in varchar2 );
```

Parameters

Table 10-3 ADD_ERROR Procedure Signature 3 Parameters

Parameters	Description
`p_error_code`	Name of shared component text message.
`p_additional_info`	Additional error information needed if the error is displayed on the error page.
`p0` through `p9`	Values for %0 through %9 placeholders defined in the text message.
`p_escape_placeholders`	If set to `TRUE`, the values provided in `p0` through `p9` are escaped with `sys.htf.escape_sc` before replacing the placeholder in the text message. If set to `FALSE`, values are not escaped.
`p_display_location`	Specifies where the error message is displayed. Use the constants defined for `p_display_location`. See "Constants and Attributes used for Result Types (page 10-1)."
`p_page_item_name`	Name of the page item on the current page that is highlighted if `apex_error.c_inline_with_field` or `apex_error.c_inline_with_field_and_notif` are used as the display location.

Example

This example illustrates how to add a custom error message, where the text is stored in a text message, to the error stack. The P5_CUSTOMER_ID item is highlighted on the page. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_error_code       => 'INVALID_CUSTOMER_ID',
    p0                 => l_customer_id,
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_page_item_name   => 'P5_CUSTOMER_ID' );
```

10.6 ADD_ERROR Procedure Signature 4

This procedure adds an error message to the error stack that is used to display an error for a tabular form inline in a notification. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_message          in varchar2,
    p_additional_info  in varchar2 default null,
    p_display_location in varchar2,
    p_region_id        in number,
    p_column_alias     in varchar2 default null,
    p_row_num          in number );
```

Parameters

Table 10-4 ADD_ERROR Procedure Signature 4 Parameters

Parameters	Description
`p_message`	Displayed error message.
`p_additional_info`	Additional error information needed if the error is displayed on the error page.
`p_display_location`	Specifies where the error message is displayed. Use the constant `apex_error.c_inline_with_field` or `apex_error.c_inline_with_field_and_notif`. See "Constants and Attributes used for Result Types (page 10-1)."
`p_region_id`	The ID of a tabular form region on the current page. The ID can be read from the view `APEX_APPLICATION_PAGE_REGIONS`.
`p_column_alias`	Name of a tabular form column alias defined for p_region_id that is highlighted if `apex_error.c_inline_with_field` or `apex_error.c_inline_with_field_and_notif` are used as a display location.
`p_row_num`	Number of the tabular form row where the error occurred.

Example

This example illustrates how to add a custom error message for a tabular form, where the column `CUSTOMER_ID` is highlighted, to the error stack. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_message          => 'Invalid Customer ID!',
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_region_id        => l_region_id,
    p_column_alias     => 'CUSTOMER_ID',
    p_row_num          => l_row_num );
```

10.7 ADD_ERROR Procedure Signature 5

This procedure adds an error message to the error stack of a tabular form that is used to display text as defined by a shared component. This error message can be displayed to all display locations. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_error_code          in varchar2,
    p0                    in varchar2 default null,
    p1                    in varchar2 default null,
    p2                    in varchar2 default null,
    p3                    in varchar2 default null,
    p4                    in varchar2 default null,
    p5                    in varchar2 default null,
    p6                    in varchar2 default null,
    p7                    in varchar2 default null,
    p8                    in varchar2 default null,
    p9                    in varchar2 default null,
    p_escape_placeholders in boolean  default true,
    p_additional_info     in varchar2 default null,
    p_display_location    in varchar2,
    p_region_id           in number,
    p_column_alias        in varchar2 default null,
    p_row_num             in number );
```

Parameters

Table 10-5 ADD_ERROR Procedure Signature 5 Parameters

Parameters	Description
`p_error_code`	Name of shared component text message.
`p0` through `p9`	Values for %0 through %9 placeholders defined in the text message.
`p_escape_placeholders`	If set to `TRUE`, the values provided in `p0` through `p9` are escaped with `sys.htf.escape_sc` before replacing the placeholder in the text message. If set to `FALSE`, values are not escaped.
`p_additional_info`	Additional error information needed if the error is displayed on the error page.

Table 10-5 (Cont.) ADD_ERROR Procedure Signature 5 Parameters

Parameters	Description
p_display_location	Specifies where the error message is displayed. Use the constants defined for p_display_location. See "Constants and Attributes used for Result Types (page 10-1)."
p_region_id	The ID of the tabular form region on the current page. The ID can be read from the view APEX_APPLICATION_PAGE_REGIONS.
p_column_alias	The name of the tabular form column alias defined for p_region_id that is highlighted if apex_error.c_inline_with_field or apex_error.c_inline_with_field_and_notif are used as a display location.
p_row_num	Number of the tabular form row where the error occurred.

Example

This example illustrates how to add a custom error message, where the text is stored in a text message, to the error stack. The CUSTOMER_ID column on the tabular form is highlighted. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_error_code       => 'INVALID_CUSTOMER_ID',
    p0                 => l_customer_id,
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_region_id        => l_region_id,
    p_column_alias     => 'CUSTOMER_ID',
    p_row_num          => l_row_num );
```

10.8 AUTO_SET_ASSOCIATED_ITEM Procedure

This procedure automatically sets the associated page item or tabular form column based on a constraint contained in p_error.ora_sqlerrm.This procedure performs the following:

- Identifies the constraint by searching for the schema.constraint pattern.
- Only supports constraints of type P, U, R and C.
- For constraints of type C (check constraints), the procedure parses the expression to identify those columns that are used in the constraints expression.
- Using those columns, the procedure gets the first visible page item or tabular form column that is based on that column and set it as associated p_error_result.page_item_name or p_error_result.column_alias.
- If a page item or tabular form column was found, p_error_result.display_location is set to apex_error.c_inline_with_field_and_notif.

Syntax

```
APEX_ERROR.AUTO_SET_ASSOCIATED_ITEM (
    p_error_result in out nocopy t_error_result,
    p_error        in            t_error );
```

Parameters

Table 10-6 AUTO_SET_ASSOCIATED_ITEM Procedure Parameters

Parameters	Description
`p_error_result`	The result variable of your error handling function.
`p_error`	The `p_error` parameter of your error handling function.

Example

See an example of how to use this procedure in "Example of an Error Handling Function (page 10-2)."

10.9 EXTRACT_CONSTRAINT_NAME Function

This function extracts a constraint name contained in `p_error.ora_sqlerrm`. The constraint must match the pattern `schema.constraint`.

Syntax

```
APEX_ERROR.EXTRACT_CONSTRAINT_NAME (
    p_error          in t_error,
    p_include_schema in boolean default false )
    return varchar2;
```

Parameters

Table 10-7 EXTRACT_CONSTRAINT_NAME Function Parameters

Parameters	Description
`p_error`	The `p_error` parameter of your error handling function.
`p_include_schema`	If set to `TRUE`, the result is prefixed with the schema name. For example, `HR.DEMO_PRODUCT_INFO_PK`. If set to `FALSE`, only the constraint name is returned.

Example

See an example of how to use this procedure in "Example of an Error Handling Function (page 10-2)."

10.10 GET_ARIA_ERROR_ATTRIBUTES Function

This function is useful for item plug-in developers, to enhance screen reader usability of your item, specifically when that item is associated with an error on a page. This function is called as part of rendering of the item, where the main form element(s) are output. The returned WAI-ARIA attributes include:

- `aria-invalid="true"` - Indicates the page item's current value is invalid. When the user is focused on the page item, the screen reader announces 'Invalid Entry' .
- `aria-describedby="[page_item_name]_error"` - This attribute value matches up with the ID of a `<div>` tag containing the item's associated error message, enabling a screen reader to announce the actual error, when the user is focused on the page item.

Note:

Because these attributes only enhance screen reader usability, attributes are returned only if the current session is running in Screen Reader mode.

Syntax

```
function get_aria_error_attributes (
    p_item_name     in varchar2 )
    return varchar2;
```

Parameters

Table 10-8 GET_ARIA_ERROR_ATTRIBUTES Function Parameters

Parameter	Description
`p_item_name`	The page item name. This value is available by using the name attribute of the `apex_plugin.t_page_item` record type, which is passed in as the 1st parameter to all item plug-in's Render Function Callback.

Example

This example shows how this function can be used, in rendering a SELECT element, during processing of the Render Function callback for an item plug-in. This function returns additional attributes, if the page item has errors associated with it and if the user is running in Screen Reader mode.

```
...
        l_name := apex_plugin.get_input_name_for_page_item(false);
        sys.htp.prn('<select name="'||l_name||'" id="'||p_item.name||'" '||
                    apex_error.get_aria_error_attributes(p_item.name)||'>');
...
```

10.11 GET_FIRST_ORA_ERROR_TEXT Function

This function returns the first ORA error message text stored in `p_error.ora_sqlerrm`. If `p_error_ora_sqlerrm` does not contain a value, `NULL` is returned.

Syntax

```
APEX_ERROR.GET_FIRST_ORA_ERROR_TEXT (
    p_error            in t_error,
    p_include_error_no in boolean default false )
    return varchar2;
```

Parameters

Table 10-9 GET_FIRST_ORA_TEXT Function Parameters

Parameters	Description
p_error	The p_error parameter of your error handling function.
p_include_error_no	If set to TRUE, ORA-xxxx is included in the returned error message. If set to FALSE, only the error message text is returned.

Example

See an example of how to use this procedure in "Example of an Error Handling Function (page 10-2)."

10.12 INIT_ERROR_RESULT Function

This function returns the t_error_result type initialized with the values stored in p_error.

Note:

This function must be used to ensure initialization is compatible with future changes to t_error_result.

Syntax

```
APEX_ERROR.INIT_ERROR_RESULT (
    p_error  in t_error)
    return   t_error_result;
```

Parameters

Table 10-10 INT_ERROR_RESULT Function Parameters

Parameters	Description
p_error	The p_error parameter of your error handling function.

Example

See an example of how to use this function in "Example of an Error Handling Function (page 10-2)."

11

APEX_INSTANCE_ADMIN

The APEX_INSTANCE_ADMIN package provides utilities for managing an Oracle Application Express runtime environment. You use the APEX_INSTANCE_ADMIN package to get and set email settings, Oracle Wallet settings, report printing settings and to manage schema to workspace mappings. APEX_INSTANCE_ADMIN can be executed by the SYS, SYSTEM, and APEX_050000 database users and any database user granted the role APEX_ADMINISTRATOR_ROLE.

TRUNCATE_LOG Procedure (page 11-24)

UNRESTRICT_SCHEMA Procedure (page 11-25)

11.1 Available Parameter Values

Table 11-1 (page 11-2) lists all the available parameter values you can set within the APEX_INSTANCE_ADMIN package, including parameters for email, wallet, and reporting printing.

Table 11-1 Available Parameters

Parameter Name	Description
ACCOUNT_LIFETIME_DAYS	The maximum number of days an end-user account password may be used before the account is expired.
ALLOW_DB_MONITOR	If set to Y, the default, database monitoring is enabled. If set to N, it is disabled.
ALLOW_HOSTNAMES	If set, users can only navigate to an application if the URL's hostname part contains this value. Instance administrators can configure more specific values at workspace level.
ALLOW_PUBLIC_FILE_UPLOAD	If set to Y, file uploads are allowed without user authentication. If set to N, the default, they are not allowed.
ALLOW_RAS	This parameter is only supported if running Oracle Database 12*c*. If set to Y, enable Real Application Security support for applications. If set to N (the default), Real Application Security cannot be used.
ALLOW_REST	If set to Y, the default, developers are allowed to expose report regions as RESTful services. If set to N, the are not allowed.
APEX_BUILDER_AUTHENTICATION	Controls the authentication scheme for the internal builder applications. Valid parameter values include: • APEX - Application Express workspace accounts authentication (default) • DB - Database accounts authentication • HEADER - HTTP header variable based authentication • SSO - Oracle Single Sign-On authentication • LDAP - LDAP authentication
APEX_REST_PATH_PREFIX	Controls the URI path prefix used to access built-in RESTful Services exposed by Application Express. For example, built-in RESTful Service for referencing static application files using #APP_IMAGES# token. If the default prefix (r) conflicts with RESTful Services defined by users, adjust this preference to avoid the conflict.
APPLICATION_ACTIVITY_LOGGING	Controls instance wide setting of application activity log ([A]lways, [N]ever, [U]se application settings)

Table 11-1 (Cont.) Available Parameters

Parameter Name	Description
APPLICATION_ID_MAX	The largest possible ID for a websheet or database application.
APPLICATION_ID_MIN	The smallest possible ID for a websheet or database application.
AUTOEXTEND_TABLESPACES	If set to Y, the default, provisioned tablespaces is autoextended up to a maximum size. If set to N tablespaces are not autoextended.
BIGFILE_TABLESPACES_ENABLED	If set to Y, the tablespaces provisioned through Oracle Application Express are created as bigfile tablespaces. If set to N, the tablespaces are created as smallfile tablespaces.
CHECKSUM_HASH_FUNCTION	Defines the algorithm that is used to create one way hashes for URL checksums.Valid values are MD5 (deprecated), SH1 (SHA-1), SH256 (SHA-2, 256 bit), SH384 (SHA-2, 384 bit), SH512 (SHA-2, 512 bit) and null. The SHA-2 algorithms are only available on Oracle Database Release 12*g* and later. A null value evaluates to the most secure algorithm available and is the default.
CHECK_FOR_UPDATES	If set to N, the check for Oracle Application Express and Oracle REST Data Services product updates is disabled for the entire instance, regardless of preferences specified by individual developers. The default is Y.
DELETE_UPLOADED_FILES_AFTER_DAYS	Uploaded files like application export files, websheet export files, spreadsheet data load files are automatically deleted after this number of days. Default is 14.
DISABLE_ADMIN_LOGIN	If set to Y, administration services are disabled. If set to N, the default, they are not disabled.
DISABLE_WORKSPACE_LOGIN	If set to Y, the workspace login is disabled. If set to N, the default, the login is not disabled.
DISABLE_WS_PROV	If set to Y, the workspace creation is disabled for requests sent out by using e-mail notification. If set to N, the default, they are not disabled.
EMAIL_IMAGES_URL	Specifies the full URL to the images directory of Application Express instance, including the trailing slash after the images directory. For example: `http://your_server/i/` This setting is used for Oracle Application Express system-generated emails.
EMAIL_INSTANCE_URL	Specifies the URL to Oracle Application Express instance, including the trailing slash after the Database Access Descriptor. For example: `http://your_server/pls/apex/` This setting used for Oracle Application Express system-generated emails.

Table 11-1 (Cont.) Available Parameters

Parameter Name	Description
ENABLE_TRANSACTIONAL_SQL	If set to Y, the default, transactional SQL commands are enabled on this instance. If set to N, they are not enabled.
ENCRYPTED_TABLESPACES_ENABLED	If set to Y, the tablespaces provisioned through Oracle Application Express are created as encrypted tablespaces. If set to N, the tablespaces are not encyrpted.
EXPIRE_FIND_USER_ACCOUNTS	If set to Y, expiration of Application Express accounts is enabled. If set to N, they are not enabled.
HTTP_ERROR_STATUS_ON_ERROR_PAGE_ENABLED	Used in conjunction with the APEX_INSTANCE_ADMIN.SET_PARAMETER procedure. If set to N, the default, Oracle Application Express presents an error page to the end user for all unhanded errors. If set to Y, returns an HTTP 400 status to the end user's client browser when the Application Express engine encounters an unhandled error.
HTTP_RESPONSE_HEADERS	List of http response headers, separated by newline (chr(10)). Application Express writes these headers on each request, before rendering the page. The substitution string #CDN# within the headers is replaced with the content delivery networks that are known to Application Express.
HTTP_STS_MAX_AGE	REQUIRE_HTTPS must be set to A for this parameter to be relevant. Application Express emits a Strict-Transport-Security header, with max-age=<value>, on HTTPS requests if HTTP_STS_MAX_AGE has a value greater than 0. If the request protocol is HTTP, instead of processing the request, Application Express redirects to a HTTPS URL.
INBOUND_PROXIES	Comma-separated list of IP addresses for proxy servers through which requests come in.
KEEP_SESSIONS_ON_UPGRADE	This flag affects application upgrades. If set to N, the default, delete sessions associated with the application. If set to Y, leave sessions unaffected.
LOGIN_THROTTLE_DELAY	The flag which determines the time increase in seconds after failed logins.
LOGIN_THROTTLE_METHODS	The methods to count failed logins. Colon-separated list of USERNAME_IP, USERNAME, IP.
MAX_SESSION_IDLE_SEC	The number of seconds an internal application may be idle.
MAX_SESSION_IDLE_SEC	The number of seconds an internal application may be idle.
MAX_SESSION_LENGTH_SEC	The number of seconds an internal application session may exist.

Table 11-1 (Cont.) Available Parameters

Parameter Name	Description
MAX_SESSION_LENGTH_SEC	The number of seconds an internal application session may exist.
PASSWORD_ALPHA_CHARACTERS	The alphabetic characters used for password complexity rules. Default list of alphabetic characters include the following: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
PASSWORD_HASH_FUNCTION	Defines the algorithm that is used to create one way hashes for workspace user passwords. Valid values are MD5 (deprecated), SH1 (SHA-1), SH256 (SHA-2, 256 bit), SH384 (SHA-2, 384 bit), SH512 (SHA-2, 512 bit) and null. The SHA-2 algorithms are only available on Oracle Database Release 12*g* and later. A null value evaluates to the most secure algorithm available and is the default.
PASSWORD_HASH_ITERATIONS	Defines the number of iterations for the PASSWORD_HASH_FUNCTION (default 10000).
PASSWORD_HISTORY_DAYS	Defines the maximum number of days a developer or administrator account password may be used before the account expires. The default value is 45 days.
PASSWORD_PUNCTUATION_CHARACTERS	The punctuation characters used for password complexity rules. Default list of punctuation characters include the following: !"#$%&()``*+,-/:;<=>?_
PLSQL_EDITING	If set to Y, the default, the SQL Workshop Object Browser is enabled to allow users to edit and compile PL/SQL. If set to N, users are not allowed.
PRINT_BIB_LICENSED	Specify either standard support or advanced support. Advanced support requires an Oracle BI Publisher license. Valid values include: • APEX_LISTENER - Requires Oracle Rest Data Services • ADVANCED - Requires Oracle BI Publisher • STANDARD
PRINT_SVR_HOST	Specifies the host address of the print server converting engine, for example, localhost. Enter the appropriate host address if the print server is installed at another location.
PRINT_SVR_PORT	Defines the port of the print server engine, for example 8888. Value must be a positive integer.
PRINT_SVR_PROTOCOL	Valid values include: • http • https

Table 11-1 (Cont.) Available Parameters

Parameter Name	Description
PRINT_SVR_SCRIPT	Defines the script that is the print server engine, for example: /xmlpserver/convert
QOS_MAX_SESSION_KILL_TIMEOUT	Number of seconds that an active old session can live, when QOS_MAX_SESSION_REQUESTS has been reached. The oldest database session with LAST_CALL_ET greater than QOS_MAX_SESSION_KILL_TIMEOUT is killed.
QOS_MAX_SESSION_REQUESTS	Number of allowed concurrent requests to one session associated with this workspace.
QOS_MAX_WORKSPACE_REQUESTS	Number of allowed concurrent requests to sessions in this workspace.
REQ_NEW_SCHEMA	If set to Y, the option for new schema for new workspace requests is enabled. If set to N, the default, the option is disabled.
REQUIRE_HTTPS	Set to A, to enforce HTTPS for the entire Application Express instance. Set to I, to enforce HTTPS within the Application Express development and administration applications. Set to N, to allow all applications to be used when the protocol is either HTTP or HTTPS. Please note developers can also enforce HTTPS at the application level, by setting the **Secure** attribute of an application scheme's cookie.
REQUIRE_HTTPS	Set to Y to allow authentication pages within the Application Express development and administration applications to be used only when the protocol is HTTPS. Select N to allow these application pages to be used when the protocol is either HTTP or HTTPS.
REQUIRE_VERIFICATION_CODE	If set to Y, the Verification Code is displayed and is required for someone to request a new workspace. If set to N, the default, the Verification Code is not required.
RESTFUL_SERVICES_ENABLED	If set to Y, the default, RESTful services development is enabled. If set to N, RESTful services are not enabled.
RM_CONSUMER_GROUP	If set, this is the resource manager consumer group to be used for all page events. A more specific group can be configured at workspace level.
SERVICE_REQUEST_FLOW	Determines default provisioning mode. Default is MANUAL.
SERVICE_REQUESTS_ENABLED	If set to Y, the default, workspace service requests for schemas, storage, and termination is enabled. If set to N, these requests are disabled.

Table 11-1 (Cont.) Available Parameters

Parameter Name	Description
SMTP_FROM	Defines the "from" address for administrative tasks that generate email, such as approving a provision request or resetting a password. Enter a valid email address, for example: `someone@somewhere.com`
SMTP_HOST_ADDRESS	Defines the server address of the SMTP server. If you are using another server as an SMTP relay, change this parameter to that server's address. Default setting: `localhost`
SMTP_HOST_PORT	Defines the port the SMTP server listens to for mail requests. Default setting: `25`
SMTP_PASSWORD	Defines the password Application Express takes to authenticate itself against the SMTP server, with the parameter SMTP_USERNAME.
SMTP_TLS_MODE	Defines whether Application Express opens an encrypted connection to the SMTP server. Encryption is only supported on database versions 11.2.0.2 and later. On earlier database versions, the connection is not encrypted. If set to N, the connection is unencrypted (default). If set to Y, the connection is encrypted before data is sent. If STARTTLS, Application Express sends the SMTP commands EHLO <SMTP_HOST_ADDRESS> and STARTTLS before encrypting the connection.
SMTP_USERNAME	Defines the username Application Express takes to authenticate itself against the SMTP server (default is null). Starting with database version 11.2.0.2, Application Express uses UTL_MAIL's AUTH procedure for authentication. This procedure negotiates an authentication mode with the SMTP server. With earlier database versions, the authentication mode is always AUTH LOGIN. If SMTP_USERNAME is null, no authentication is used.
SQL_SCRIPT_MAX_OUTPUT_SIZE	The maximum allowable size for an individual script result. Default is 200000.
SSO_LOGOUT_URL	Defines the URL Application Express redirects to in order to trigger a logout from the Single Sign-On server. Application Express automatically appends "?p_done_url=...login url...". Example: `https://login.mycompany.com/pls/orasso/orasso.wwsso_app_admin.ls_logout`

Table 11-1 (Cont.) Available Parameters

Parameter Name	Description
STRONG_SITE_ADMIN_PASSWORD	If set to Y, the default, the apex_admin password must conform to the default set of strong complexity rules. If set to N, the password is not required to follow the strong complexity rules.
SYSTEM_HELP_URL	Location of the help and documentation accessed from the Help link within the development environment. Default is `http://apex.oracle.com/doc41`.
TRACING_ENABLED	If set to Y (the default), an application with Debug enabled can also generate server side db trace files using `&p_trace=YES` on the URL. If set to N, the request to create a trace file is ignored.
USERNAME_VALIDATION	The regular expression used to validate a username if the Builder authentication scheme is not APEX. Default is as follows: ^[[:alnum:]._%-]+@[[:alnum:].-]+\.[[:alpha:]]{2,4}$
WALLET_PATH	The path to the wallet on the file system, for example: `file:/home/<username>/wallets`
WALLET_PWD	The password associated with the wallet.
WEBSHEET_SQL_ACCESS	If set to Y, the default, SQL tags and SQL reports are possible in Websheet applications. If set to N, they are not possible.
WORKSPACE_EMAIL_MAXIMUM	Maximum number of emails allowed to be sent by using APEX_MAIL per workspace in a 24 hour period. Default is `1000`.
WORKSPACE_MAX_FILE_BYTES	The maximum number of bytes for uploaded files for a workspace. A setting at the workspace-level overrides the instance-level setting.
WORKSPACE_MAX_OUTPUT_SIZE	The maximum space allocated for script results. Default is `2000000`.
WORKSPACE_PROVISION_DEMO_OBJECTS	If set to Y, the default, demonstration applications and database objects are created in new workspaces. If set to N, they are not created in the current workspace.
WORKSPACE_WEBSHEET_OBJECTS	If set to Y, the default, Application Express Websheet database objects are created in new workspaces. If set to N, they are not created in the current workspace.

See Also:

- "Configuring Email in a Runtime Environment"
- "Configuring Wallet Information"
- "Configuring Report Printing Settings in a Runtime Environment"
- *Oracle Application Express Administration Guide*

11.2 ADD_SCHEMA Procedure

The ADD_SCHEMA procedure adds a schema to a workspace to schema mapping.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_SCHEMA(
    p_workspace    IN VARCHAR2,
    p_schema       IN VARCHAR2);
```

Parameters

Table 11-2 ADD_SCHEMA Parameters

Parameter	Description
p_workspace	The name of the workspace to which the schema mapping is added.
p_schema	The schema to add to the schema to workspace mapping.

Example

The following example demonstrates how to use the ADD_SCHEMA procedure to map a schema mapped to a workspace.

```
BEGIN
    APEX_INSTANCE_ADMIN.ADD_SCHEMA('MY_WORKSPACE','FRANK');
END;
```

11.3 ADD_WORKSPACE Procedure

The ADD_WORKSPACE procedure adds a workspace to an Application Express Instance.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_WORKSPACE(
    p_workspace_id        IN NUMBER DEFAULT NULL,
    p_workspace           IN VARCHAR2,
    p_source_identifier   IN VARCHAR2 DEFAULT NULL,
    p_primary_schema      IN VARCHAR2,
    p_additional_schemas  IN VARCHAR2,
    p_rm_consumer_group   IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 11-3 ADD_WORKSPACE Parameters

Parameter	Description
p_workspace_id	The ID to uniquely identify the workspace in an Application Express instance. This may be left null and a new unique ID is assigned.
p_workspace	The name of the workspace to be added.
p_source_identifier	A short identifier for the workspace used when synchronizing feedback between different instances.
p_primary_schema	The primary database schema to associate with the new workspace.
p_additional_schemas	A colon delimited list of additional schemas to associate with this workspace.
p_rm_consumer_group	Resource Manager consumer group which is used when executing applications of this workspace.

Example

The following example demonstrates how to use the ADD_WORKSPACE procedure to add a new workspace named MY_WORKSPACE using the primary schema, SCOTT, along with additional schema mappings for HR and OE.

```
BEGIN
    APEX_INSTANCE_ADMIN.ADD_WORKSPACE (
        p_workspace_id       => 8675309,
        p_workspace          => 'MY_WORKSPACE',
        p_primary_schema     => 'SCOTT',
        p_additional_schemas => 'HR:OE' );
END;
```

11.4 CREATE_SCHEMA_EXCEPTION Procedure

This procedure creates an exception which allows assignment of a restricted schema to a specific workspace.

Syntax

```
APEX_INSTANCE_ADMIN.CREATE_SCHEMA_EXCEPTION (
    p_schema    in varchar2,
    p_workspace in varchar2 );
```

Parameter

Table 11-4 CREATE_SCHEMA_EXCPETION Parameters

Parameter	Description
p_schema	The schema.

Table 11-4 (Cont.) CREATE_SCHEMA_EXCPETION Parameters

Parameter	Description
p_workspace	The workspace.

Example

This example allows the assignment of restricted schema HR to workspace HR_WORKSPACE.

```
begin
    apex_instance_admin.create_schema_exception (
        p_schema    => 'HR',
        p_workspace => 'HR_WORKSPACE' );
    commit;
end;
```

See Also:

- "RESTRICT_SCHEMA Procedure (page 11-21)"
- "UNRESTRICT_SCHEMA Procedure (page 11-25)"
- "REMOVE_SCHEMA_EXCEPTION Procedure (page 11-16)"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure (page 11-17)"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure (page 11-19)"

11.5 FREE_WORKSPACE_APP_IDS Procedure

This procedure removes the reservation of application IDs for a given workspace ID. Use this procedure to undo a reservation, when the reservation is not necessary anymore because it happened by mistake or the workspace no longer exists. To reserve application IDs for a given workspace, see "RESERVE_WORKSPACE_APP_IDS Procedure (page 11-19)."

Syntax

```
APEX_INSTANCE_ADMIN.FREE_WORKSPACE_APP_IDS (
    p_workspace_id IN NUMBER );
```

Parameters

Table 11-5 FREE_WORKSPACE_APP_IDS Parameters

Parameter	Description
p_workspace_id	The unique ID of the workspace.

Example

This example illustrates how to undo the reservation of application IDS that belong to a workspace with an ID of 1234567890.

```
begin
    apex_instance_admin.free_workspace_app_ids(1234567890);
end;
```

11.6 GET_PARAMETER Function

The GET_PARAMETER function retrieves the value of a parameter used in administering a runtime environment.

Syntax

```
APEX_INSTANCE_ADMIN.GET_PARAMETER(
    p_parameter     IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 11-6 GET_PARAMETER Parameters

Parameter	Description
p_parameter	The instance parameter to be retrieved. See "Available Parameter Values (page 11-2)".

Example

The following example demonstrates how to use the GET_PARAMETER function to retrieve the SMTP_HOST_ADDRESS parameter currently defined for an Oracle Application Express instance.

```
DECLARE
    L_VAL VARCHAR2(4000);
BEGIN
    L_VAL :=APEX_INSTANCE_ADMIN.GET_PARAMETER('SMTP_HOST_ADDRESS');
    DBMS_OUTPUT.PUT_LINE('The SMTP Host Setting Is: '||L_VAL);
END;
```

11.7 GET_SCHEMAS Function

The GET_SCHEMAS function retrieves a comma-delimited list of schemas that are mapped to a given workspace.

Syntax

```
APEX_INSTANCE_ADMIN.GET_SCHEMAS(
    p_workspace     IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 11-7 GET_SCHEMAS Parameters

Parameter	Description
p_workspace	The name of the workspace from which to retrieve the schema list.

Example

The following example demonstrates how to use the GET_SCHEMA function to retrieve the underlying schemas mapped to a workspace.

```
DECLARE
    L_VAL VARCHAR2(4000);
BEGIN
    L_VAL :=APEX_INSTANCE_ADMIN.GET_SCHEMAS('MY_WORKSPACE');
    DBMS_OUTPUT.PUT_LINE('The schemas for my workspace: '||L_VAL);
END;
```

11.8 GET_WORKSPACE_PARAMETER

The GET_WORKSPACE_PARAMETER procedure gets the workspace parameter.

Syntax

```
get_workspace_parameter(
    p_workspace     IN VARCHAR2,
    p_parameter     IN VARCHAR2,
```

Parameters

Table 11-8 GET_WORKSPACE_PARAMETER Parameters

Parameter	Description
p_workspace	The name of the workspace to which you are getting the workspace parameter.
p_parameter	The parameter name that overrides the instance parameter value of the same name for this workspace. Parameter names include: • ALLOW_HOSTNAMES • MAX_SESSION_IDLE_SEC • MAX_SESSION_LENGTH_SEC • QOS_MAX_WORKSPACE_REQUESTS • QOS_MAX_SESSION_REQUESTS • QOS_MAX_SESSION_KILL_TIMEOUT • RM_CONSUMER_GROUP • WORKSPACE_EMAIL_MAXIMUM • WORKSPACE_MAX_FILE_BYTES

Example

The following example prints the value of ALLOW_HOSTNAMES for the HR workspace.

```
BEGIN
    DBMS_OUTPUT.PUT_LINE (
APEX_INSTANCE_ADMIN.GET_WORKSPACE_PARAMETER (
            p_workspace => 'HR',
            p_parameter => 'ALLOW_HOSTNAMES' ));
END;
```

11.9 REMOVE_APPLICATION Procedure

The REMOVE_APPLICATION procedure removes the application specified from the Application Express instance.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_APPLICATION (
    p_application_id IN NUMBER);
```

Parameters

Table 11-9 REMOVE_APPLICATION Parameters

Parameter	Description
p_application_id	The ID of the application to remove.

Example

The following example demonstrates how to use the REMOVE_APPLICATION procedure to remove an application with an ID of 100 from an Application Express instance.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_APPLICATION(100);
END;
```

11.10 REMOVE_SAVED_REPORT Procedure

The REMOVE_SAVED_REPORT procedure removes a specific user's saved interactive report settings for a particular application.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(
    p_application_id     IN NUMBER,
    p_report_id          IN NUMBER);
```

Parameters

Table 11-10 REMOVE_SAVED_REPORT Parameters

Parameter	Description
p_application_id	The ID of the application for which to remove user saved interactive report information.
p_report_id	The ID of the saved user interactive report to be removed.

Example

The following example demonstrates how to use the REMOVE_SAVED_REPORT procedure to remove user saved interactive report with the ID 123 for the application with an ID of 100.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(100,123);
END;
```

11.11 REMOVE_SAVED_REPORTS Procedure

The REMOVE_SAVED_REPORTS procedure removes all user saved interactive report settings for a particular application or for the entire instance.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(
    p_application_id     IN NUMBER DEFAULT NULL);
```

Parameters

Table 11-11 REMOVE_SAVED_REPORTS Parameters

Parameter	Description
p_application_id	The ID of the application for which to remove user saved interactive report information. If this parameter is left null, all user saved interactive reports for the entire instance is removed.

Example

The following example demonstrates how to use the REMOVE_SAVED_REPORTS procedure to remove user saved interactive report information for the application with an ID of 100.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(100);
END;
```

11.12 REMOVE_SCHEMA Procedure

This REMOVE_SCHEMA procedure removes a workspace to schema mapping.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA(
    p_workspace     IN VARCHAR2,
    p_schema        IN VARCHAR2);
```

Parameters

Table 11-12 REMOVE_SCHEMA Parameters

Parameter	Description
p_workspace	The name of the workspace from which the schema mapping is removed.
p_schema	The schema to remove from the schema to workspace mapping.

Example

The following example demonstrates how to use the REMOVE_SCHEMA procedure to remove the schema named Frank from the MY_WORKSPACE workspace to schema mapping.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SCHEMA('MY_WORKSPACE','FRANK');
END;
```

11.13 REMOVE_SCHEMA_EXCEPTION Procedure

This procedure removes an exception that allows the assignment of a restricted schema to a given workspace.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA_EXCEPTION (
    p_schema    in varchar2,
    p_workspace in varchar2 );
```

Parameter

Table 11-13 REMOVE_SCHEMA_EXCEPTION Parameters

Parameter	Description
p_schema	The schema.
p_workspace	The workspace.

Example

This example removes the exception that allows the assignment of schema HR to workspace HR_WORKSPACE.

```
begin
    apex_instance_admin.remove_schema_exception (
        p_schema    => 'HR',
        p_workspace => 'HR_WORKSPACE' );
```

```
    commit;
end;
```

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure (page 11-10)"
- "RESTRICT_SCHEMA Procedure (page 11-21)"
- "UNRESTRICT_SCHEMA Procedure (page 11-25)"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure (page 11-17)"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure (page 11-19)"

11.14 REMOVE_SCHEMA_EXCEPTIONS Procedure

This procedure removes all exceptions that allow the assignment of a given schema to workspaces.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA_EXCEPTIONS (
      p_schema in varchar2 );
```

Parameter

Table 11-14 REMOVE_SCHEMA_EXCEPTIONS Parameter

Parameter	Description
p_schema	The schema.

Example

This example removes all exceptions that allow the assignment of the HR schema to workspaces.

```
begin
    apex_instance_admin.remove_schema_exceptions (
            p_schema => 'HR' );
    commit;
end;
```

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure (page 11-10)"
- "RESTRICT_SCHEMA Procedure (page 11-21)"
- "UNRESTRICT_SCHEMA Procedure (page 11-25)"
- "REMOVE_SCHEMA_EXCEPTION Procedure (page 11-16)"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure (page 11-19)"

11.15 REMOVE_SUBSCRIPTION Procedure

The REMOVE_SUBSCRIPTION procedure removes a specific interactive report subscription.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SUBSCRIPTION(
    p_subscription_id     IN NUMBER);
```

Parameters

Table 11-15 REMOVE_SUBSCRIPTION Procedure Parameters

Parameter	Description
p_subscription_id	The ID of the interactive report subscription to be removed.

Example

The following example demonstrates how to use the REMOVE_SUBSCRIPTION procedure to remove interactive report subscription with the ID 12345. Use of APEX_APPLICATION_PAGE_IR_SUB view can help identifying the subscription ID to remove.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SUBSCRIPTION (
        p_subscription_id => 12345);
END;
```

11.16 REMOVE_WORKSPACE Procedure

The REMOVE_WORKSPACE procedure removes a workspace from an Application Express instance.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE(
    p_workspace         IN VARCHAR2,
    p_drop_users        IN VARCHAR2 DEFAULT 'N',
    p_drop_tablespaces  IN VARCHAR2 DEFAULT 'N' );
```

Parameters

Table 11-16 REMOVE_WORKSPACE Parameters

Parameter	Description
p_workspace	The name of the workspace to be removed.
p_drop_users	'Y' to drop the database user associated with the workspace. The default is 'N'.
p_drop_tablespaces	'Y' to drop the tablespace associated with the database user associated with the workspace. The default is 'N'.

Example

The following example demonstrates how to use the REMOVE_WORKSPACE procedure to remove an existing workspace named MY_WORKSPACE, along with the associated database users and tablespace.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE('MY_WORKSPACE','Y','Y');
END;
```

11.17 REMOVE_WORKSPACE_EXCEPTIONS Procedure

This procedure removes all exceptions that allow the assignment of restricted schemas to given workspace.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE_EXCEPTIONS (      p_workspace in varchar2 );
```

Parameter

Table 11-17 REMOVE_WORKSPACE_EXCEPTIONS Parameter

Parameter	Description
p_workspace	The workspace.

Example

This example removes all exceptions that allow the assignment of restricted schemas to HR_WORKSPACE.

```
begin    apex_instance_admin.remove_schema_exceptions (            p_workspace =>
'HR_WORKSPACE' );    commit;end;
```

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure (page 11-10)"
- "RESTRICT_SCHEMA Procedure (page 11-21)"
- "UNRESTRICT_SCHEMA Procedure (page 11-25)"
- "REMOVE_SCHEMA_EXCEPTION Procedure (page 11-16)"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure (page 11-17)"

11.18 RESERVE_WORKSPACE_APP_IDS Procedure

This procedure permanently reserves the IDs of websheet and database applications in a given workspace. Even if the workspace and its applications get removed, developers can not create other applications with one of these IDs.

Syntax

```
APEX_INSTANCE_ADMIN.RESERVE_WORKSPACE_APP_IDS (
    p_workspace_id IN NUMBER );
```

Parameters

Table 11-18 RESERVE_WORKSPACE_APP_IDS Parameters

Parameter	Description
p_workspace_id	The unique ID of the workspace.

Example

This example demonstrates setting up two separate Application Express instances where the application IDs are limited to within a specific range. At a later point, a workspace and all of it's applications are moved from instance 1 to instance 2. For the workspace that is moved, the developer reserves all of its application IDs to ensure that no applications with the same IDs are created on instance 1.

1. After setting up Application Express instance 1, ensure that application IDs are between 100000 and 199999.

   ```
   begin
       apex_instance_admin.set_parameter('APPLICATION_ID_MIN', 100000);
       apex_instance_admin.set_parameter('APPLICATION_ID_MAX', 199999);
   end;
   ```

2. After setting up Application Express instance 2, ensure that application IDs are between 200000 and 299999.

   ```
   begin
       apex_instance_admin.set_parameter('APPLICATION_ID_MIN', 200000);
       apex_instance_admin.set_parameter('APPLICATION_ID_MAX', 299999);
   end;
   ```

3. Later, the operations team decides that workspace MY_WORKSPACE with ID 1234567890 should be moved from instance 1 to instance 2. The required steps are:

   a. Export the workspace, applications and data on instance 1 (not shown here).

   b. Ensure that no other application on instance 1 can reuse application IDs of this workspace.

      ```
      begin
          apex_instance_admin.reserve_workspace_app_ids(1234567890);
      end;
      ```

   c. Drop workspace, accompanying data and users on instance 1.

      ```
      begin
          apex_instance_admin.remove_workspace('MY_WORKSPACE');
      end;
      ```

   d. Import the workspace, applications and data on instance 2 (not shown here).

See Also:

To undo a reservation, see "FREE_WORKSPACE_APP_IDS Procedure (page 11-11)."

11.19 RESTRICT_SCHEMA Procedure

This procedure revokes the privilege to assign a schema to workspaces.

Syntax

```
APEX_INSTANCE_ADMIN.RESTRICT_SCHEMA (
    p_schema in varchar2 );
```

Parameter

Table 11-19 RESTRICT_SCHEMA Parameters

Parameter	Description
p_schema	The schema.

Example

This example revokes the privilege to assign schema HR to workspaces.

```
begin
    apex_instance_admin.restrict_schema(p_schema => 'HR');
    commit;
end;
```

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure (page 11-10)"
- "UNRESTRICT_SCHEMA Procedure (page 11-25)"
- "REMOVE_SCHEMA_EXCEPTION Procedure (page 11-16)"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure (page 11-17)"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure (page 11-19)"

11.20 SET_LOG_SWITCH_INTERVAL Procedure

Set the log switch interval for each of the logs maintained by Application Express.

Syntax

```
APEX_INSTANCE_ADMIN.SET_LOG_SWITCH_INTERVAL(
    p_log_name              IN VARCHAR2,
    p_log_switch_after_days IN NUMBER );
```

Parameters

Table 11-20 SET_LOG_SWITCH_INTERVAL Parameters

Parameters	Description
p_log_name	Specifies the name of the log. Valid values include ACCESS, ACTIVITY, CLICKTHRU, and DEBUG.
p_log_switch_after_days	This interval must be a positive integer between 1 and 180.

Example

This example sets the log switch interval for the ACTIVITY log to 30 days.

```
begin
    apex_instance_admin.set_log_switch_interval( p_log_name in 'ACTIVITY',
p_log_switch_after_days => 30 );
    commit;
end;
```

11.21 SET_WORKSPACE_PARAMETER

The SET_WORKSPACE_PARAMETER procedure sets the designated workspace parameter.

Syntax

```
SET_WORKSPACE_PARAMETER(
    p_workspace     IN VARCHAR2,
    p_parameter     IN VARCHAR2,
    p_value         IN VARCHAR2 );
```

Parameters

Table 11-21 SET_WORKSPACE_PARAMETER Parameters

Parameter	Description
p_workspace	The name of the workspace to which you are setting the workspace parameter.
p_parameter	The parameter name which overrides the instance parameter value of the same for this workspace. Parameter names include: • ALLOW_HOSTNAMES • MAX_SESSION_IDLE_SEC • MAX_SESSION_LENGTH_SEC • QOS_MAX_WORKSPACE_REQUESTS • QOS_MAX_SESSION_REQUESTS • QOS_MAX_SESSION_KILL_TIMEOUT • RM_CONSUMER_GROUP • WORKSPACE_EMAIL_MAXIMUM • WORKSPACE_MAX_FILE_BYTES
p_value	The parameter value.

Example

The following example demonstrates how to use the `set_workspace_parameter` procedure to restrict URLs for accessing applications in the HR workspace that have hr.example.com in the hostname or domain name.

```
BEGIN
apex_instance_admin.set_workspace_parameter (
            p_workspace => 'HR',
            p_parameter => 'ALLOW_HOSTNAMES' ));
            p_value     => 'hr.example.com' );
END;
```

11.22 SET_PARAMETER Procedure

The SET_PARAMETER procedure sets a parameter used in administering a runtime environment. You must issue a commit for the parameter change to take affect.

Syntax

```
APEX_INSTANCE_ADMIN.SET_PARAMETER(
    p_parameter     IN VARCHAR2,
    p_value         IN VARCHAR2 DEFAULT 'N');
```

Parameters

Table 11-22 SET_PARAMETER Parameters

Parameter	Description
p_parameter	The instance parameter to be set.
p_value	The value of the parameter. See "Available Parameter Values (page 11-2)".

Example

The following example demonstrates how to use the SET_PARAMETER procedure to set the SMTP_HOST_ADDRESS parameter for an Oracle Application Express instance.

```
BEGIN
    APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_HOST_ADDRESS', 'mail.example.com');
    COMMIT;
END;
```

11.23 SET_WORKSPACE_CONSUMER_GROUP Procedure

The SET_WORKSPACE_CONSUMER_GROUP procedure sets a Resource Manager Consumer Group to a workspace.

Syntax

```
set_workspace_consumer_group(
    p_workspace in varchar2,
    p_rm_consumer_group in varchar2 );
```

Parameters

Table 11-23 SET_WORKSPACE_CONSUMER_GROUP Parameters

Parameters	Description
p_workspace	This is the name of the workspace for which the resource consumer group is to be set.
p_rm_consumer_group	The parameter P_RM_CONSUMER_GROUP is the Oracle Database Resource Manager Consumer Group name. The consumer group does not have to exist at the time this procedure is invoked. But if the Resource Manager Consumer Group is set for a workspace and the consumer group does not exist, then an error will be raised when anyone attempts to login to this workspace or execute any application in the workspace. If the value of P_RM_CONSUMER_GROUP is null, then the Resource Manager consumer group associated with the specified workspace is cleared.

Example

The following example sets the workspace to the Resource Manager consumer group "CUSTOM_GROUP1":

```
begin
        apex_instance_admin.set_workspace_consumer_group(
        p_workspace => 'MY_WORKSPACE',
        p_rm_consumer_group => 'CUSTOM_GROUP1' );
    commit;
end;
/
```

11.24 TRUNCATE_LOG Procedure

The TRUNCATE_LOG procedure truncates the log entries specified by the input parameter.

Syntax

```
APEX_INSTANCE_ADMIN.TRUNCATE_LOG(
    p_log     IN VARCHAR2);
```

Parameters

Table 11-24 TRUNCATE_LOG Parameters

Parameter	Description
p_log	This parameter can have one of the following values: ACTIVITY - removes all entries that record page access. USER_ACCESS - removes all entries that record user login. MAIL - removes all entries that record mail sent. CLICKS - removes all entries that record clicks tracked to external sites. LOCK_INSTALL_SCRIPT - removes all entries that record developer locking of supporting objects script. LOCK_PAGE - removes all entries that record developer locking of pages. WORKSPACE_HIST - removes all entries that record daily workspace summary. PURGE - removes all entries that record automatic workspace purge activity. FILE - removes all entries that record automatic file purge activity. SCRIPT - removes all entries that record results of SQL scripts executed in SQL Workshop. SQL - removes all entries that record the history of commands executed in SQL Workshop SQL Commands

Example

The following example demonstrates how to use the TRUNCATE_LOG procedure to remove all log entries that record access to Application Express application pages.

```
BEGIN
  APEX_INSTANCE_ADMIN.TRUNCATE_LOG('ACTIVITY');
END;
```

11.25 UNRESTRICT_SCHEMA Procedure

This procedure re-grants the privilege to assign a schema to workspaces, if it has been revoked before.

Syntax

```
APEX_INSTANCE_ADMIN.UNRESTRICT_SCHEMA (
    p_schema in varchar2 );
```

Parameter

Table 11-25 RESTRICT_SCHEMA Parameters

Parameter	Description
p_schema	The schema.

Example

This example re-grants the privilege to assign schema HR to workspaces.

```
begin
    apex_instance_admin.unrestrict_schema(p_schema => 'HR');
    commit;
end;
```

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure (page 11-10)"
- "RESTRICT_SCHEMA Procedure (page 11-21)"
- "REMOVE_SCHEMA_EXCEPTION Procedure (page 11-16)"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure (page 11-17),"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure (page 11-19)"

12

APEX_IR

The APEX_IR package provides utilities you can use when programming in the Oracle Application Express environment related to interactive reports. You can use the APEX_IR package to get an interactive report runtime query, add filters, reset or clear report settings, delete saved reports and manage subscriptions.

12.1 ADD_FILTER Procedure Signature 1

This procedure creates a filter on an interactive report using a report ID.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.ADD_FILTER(
    p_page_id       IN NUMBER,
    p_region_id     IN NUMBER,
    p_report_column IN VARCHAR2,
    p_filter_value  IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_report_id     IN NUMBER DEFAULT NULL);
```

Parameters

Table 12-1 ADD_FILTER Procedure Signature 1 Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region (ID).
p_report_column	Name of the report SQL column, or column alias, to be filtered.
p_filter_value	The filter value. This value is not used for N and NN.
p_operator_abbr	Filter type. Valid values are as follows: EQ = Equals NEQ = Not Equals LT = Less than LTE = Less then or equal to GT = Greater Than GTE = Greater than or equal to LIKE = SQL Like operator NLIKE = Not Like N = Null NN = Not Null C = Contains NC = Not Contains IN = SQL In Operator NIN = SQL Not In Operator
p_report_id	The saved report ID within the current application page. If p_report_id is null, it adds the filter to the last viewed report settings.

Example

The following example shows how to use the ADD_FILTER procedure to filter the interactive report with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_IR.ADD_FILTER(
        p_page_id       => 1,
```

```
        p_region_id     => 2505704029884282,
        p_report_column => 'DEPTNO',
        p_filter_value  => '30',
        p_operator_abbr => 'EQ',
        p_report_id     => 880629800374638220);
END;
```

12.2 ADD_FILTER Procedure Signature 2

This procedure creates a filter on an interactive report using a report alias.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.ADD_FILTER(
    p_page_id       IN NUMBER,
    p_region_id     IN NUMBER,
    p_report_column IN VARCHAR2,
    p_filter_value  IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_report_alias  IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 12-2 ADD_FILTER Procedure Signature 2

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region (ID).
p_report_column	Name of the report SQL column, or column alias, to be filtered.
p_filter_value	This is the filter value. This value is not used for N and NN.

Table 12-2 (Cont.) ADD_FILTER Procedure Signature 2

Parameter	Description
p_operator_abbr	Filter type. Valid values are as follows: EQ = Equals NEQ = Not Equals LT = Less than LTE = Less then or equal to GT = Greater Than GTE = Greater than or equal to LIKE = SQL Like operator NLIKE = Not Like N = Null NN = Not Null C = Contains NC = Not Contains IN = SQL In Operator NIN = SQL Not In Operator
p_report_alias	The saved report alias within the current application page. If p_report_alias is null, it adds filter to the last viewed report settings.

Example

The following example shows how to use the ADD_FILTER procedure to filter an interactive report with a report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_IR.ADD_FILTER(
        p_page_id       => 1,
        p_region_id     => 2505704029884282,
        p_report_column => 'DEPTNO',
        p_filter_value  => '30',
        p_operator_abbr => 'EQ',
        p_report_alias  => 'CATEGORY_REPORT');
END;
```

12.3 CHANGE_SUBSCRIPTION_EMAIL Procedure

This procedure changes interactive report subscriptions email address. When an email is sent out, the subscription sends message to the defined email address.

Syntax

```
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (
    p_subscription_id    IN NUMBER,
    p_email_address      IN VARCHAR2);
```

Parameters

Table 12-3 CHANGE_SUBSCRIPTION_EMAIL Parameters

Parameter	Description
p_subscription_id	Subscription ID to change the email address within the current workspace.
p_email_address	The new email address to change to. The email address needs to be a valid email syntax and cannot be set to null.

Example

The following example shows how to use CHANGE_SUBSCRIPTION_EMAIL procedure to change the email address to some.user@somecompany.com for the interactive report subscription 956136850459718525.

```
BEGIN
    APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (
        p_subscription_id => 956136850459718525,
        p_email_address => 'some.user@somecompany.com');
END;
```

12.4 CHANGE_REPORT_OWNER Procedure

This procedure changes the owner of a saved interactive report using a report ID. This procedure cannot change the owner of default interactive reports.

Syntax

```
APEX_IR.CHANGE_REPORT_OWNER (
    p_report_id    in number,
    p_old_owner    in varchar2,
    p_new_owner    in varchar2);
```

Parameters

Table 12-4 CHANGE_REPORT_OWNER Procedure

Parameters	Description
p_report_id	The saved report ID within the current application page.
p_old_owner	The previous owner name to change from (case sensitive). The owner needs to a valid login user accessing the report.
p_new_owner	The new owner name to change to (case sensitive). The owner must be a valid login user accessing the report.

Example

This example shows how to use CHANGE_REPORT_OWNER procedure to change the old owner name of *JOHN* to the new owner name of *JOHN.DOE* for a saved report. The saved report has a report ID of 1235704029884282.

```
BEGIN
    APEX_IR.CHANGE_REPORT_OWNER (
        p_report_id    => 1235704029884282,
        p_old_owner    => 'JOHN',
        p_new_owner    => 'JOHN.DOE');
END;
```

12.5 CHANGE_SUBSCRIPTION_EMAIL Procedure

This procedure changes interactive report subscriptions email address. When an email is sent out, the subscription sends message to the defined email address.

Syntax

```
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (
    p_subscription_id    IN NUMBER,
    p_email_address      IN VARCHAR2);
```

Parameters

Table 12-5 CHANGE_SUBSCRIPTION_EMAIL Parameters

Parameter	Description
`p_subscription_id`	Subscription ID to change the email address within the current workspace.
`p_email_address`	The new email address to change to. The email address needs to be a valid email syntax and cannot be set to null.

Example

The following example shows how to use `CHANGE_SUBSCRIPTION_EMAIL` procedure to change the email address to `some.user@somecompany.com` for the interactive report subscription 956136850459718525.

```
BEGIN
    APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (
        p_subscription_id => 956136850459718525,
        p_email_address => 'some.user@somecompany.com');
END;
```

12.6 CHANGE_SUBSCRIPTION_LANG Procedure

This procedure changes the interactive report subscription language.

Syntax

```
APEX_IR.CHANGE_SUBSCRIPTION_LANG(
    p_subscription_id IN NUMBER,
    p_language        IN VARCHAR2);
```

Parameters

Table 12-6 CHANGE_SUBSCRIPTION_LANG Procedure Parameters

Parameter	Description
p_subscription_id	Subscription ID to change the language within the current workspace.
p_language	This is an IANA language code. Some examples include: en, de, de-at, zh-cn, and pt-br.

Example

The following example shows how to use the CHANGE_SUBSCRIPTION_LANG procedure to change the subscription with the ID of 567890123 to German in the current workspace.

```
BEGIN
    APEX_IR.CHANGE_SUBSCRIPTION_LANG(
        p_subscription_id => 567890123,
        p_language        => 'de');
END;
```

12.7 CLEAR_REPORT Procedure Signature 1

This procedure clears report settings using the report ID.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.CLEAR_REPORT(
    p_page_id   IN NUMBER,
    p_region_id IN NUMBER,
    p_report_id IN NUMBER DEFAULT NULL);
```

Parameters

Table 12-7 CLEAR_REPORT Procedure Signature 1 Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region (ID).

Table 12-7 (Cont.) CLEAR_REPORT Procedure Signature 1 Parameters

Parameter	Description
p_report_id	The saved report ID within the current application page. If p_report_id is null, it clears the last viewed report settings.

Example

The following example shows how to use the CLEAR_REPORT procedure to clear interactive report settings with a report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_id    => 880629800374638220);
END;
```

12.8 CLEAR_REPORT Procedure Signature 2

This procedure clears report settings using report alias.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.CLEAR_REPORT(
    p_page_id      IN NUMBER,
    p_region_id    IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 12-8 CLEAR_REPORT Procedure Signature 2 Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region (ID).

Table 12-8 (Cont.) CLEAR_REPORT Procedure Signature 2 Parameters

Parameter	Description
p_report_alias	The saved report alias within the current application page. If p_report_alias is null, it clears the last viewed report settings.

Example

The following example shows how to use the CLEAR_REPORT procedure to clear interactive report settings with report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_alias => 'CATEGORY_REPORT');
END;
```

12.9 DELETE_REPORT Procedure

This procedure deletes saved interactive reports. It deletes a specific saved report in the current logged in workspace and application.

Syntax

```
APEX_IR.DELETE_REPORT(
    p_report_id IN NUMBER);
```

Parameters

Table 12-9 DELETE_REPORT Procedure Parameters

Parameter	Description
p_report_id	Report ID to delete within the current Application Express application.

Example

The following example shows how to use the DELETE_REPORT procedure to delete the saved interactive report with ID of 880629800374638220 in the current application.

```
BEGIN
    APEX_IR.DELETE_REPORT (
        p_report_id => 880629800374638220);
END;
```

12.10 DELETE_SUBSCRIPTION Procedure

This procedure deletes interactive report subscriptions.

Syntax

```
APEX_IR.DELETE_SUBSCRIPTION(
    p_subscription_id IN NUMBER);
```

Parameters

Table 12-10 DELETE_SUBSCRIPTION Procedure Parameters

Parameter	Description
p_subscription_id	Subscription ID to delete within the current workspace.

Example

The following example shows how to use the DELETE_SUBSCRIPTION procedure to delete the subscription with ID of 567890123 in the current workspace.

```
BEGIN
    APEX_IR.DELETE_SUBSCRIPTION(
        p_subscription_id => 567890123);
END;
```

12.11 GET_LAST_VIEWED_REPORT_ID Function

This function returns the last viewed base report ID of the specified page and region.

Syntax

```
APEX_IR.GET_LAST_VIEWED_REPORT_ID(
    p_page_id   IN NUMBER,
    p_region_id IN NUMBER);
```

Parameters

Table 12-11 GET_LAST_VIEWED_REPORT_ID Function Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region ID.

Example

The following example shows how to use the GET_LAST_VIEWED_REPORT_ID function to retrieve the last viewed report ID in page 1, region 2505704029884282 of the current application.

```
DECLARE
    l_report_id number;
BEGIN
    l_report_id := APEX_IR.GET_LAST_VIEWED_REPORT_ID (
        p_page_id   => 1,
        p_region_id => 2505704029884282);
END;
```

12.12 GET_REPORT Function

This function returns an interactive report runtime query.

Syntax

```
APEX_IR.GET_REPORT(
    p_page_id   IN NUMBER,
    p_region_id IN NUMBER,
    p_report_id IN NUMBER DEFAULT NULL,
    p_view      IN VARCHAR2 DEFAULT C_VIEW_REPORT );
```

Parameters

Table 12-12 GET_REPORT Function Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region ID.
p_report_id	The saved report ID within the current application page. If p_report_id is null, it gets last viewed report query.
p_view	The view type available for the report. The values can be APEX_IR.C_VIEW_REPORT, APEX_IR.C_VIEW_GROUPBY or APEX_IR.C_VIEW_PIVOT. If p_view is null, it gets the view currently used by the report. If p_view passed which doesn't exist for the current report, an error is raised.

Example 1

The following example shows how to use the GET_REPORT function to retrieve the runtime report query with bind variable information with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
DECLARE
   l_report apex_ir.t_report;
   l_query varchar2(32767);
BEGIN
    l_report := APEX_IR.GET_REPORT (
                    p_page_id => 1,
                    p_region_id => 2505704029884282,
                    p_report_id => 880629800374638220);
    l_query := l_report.sql_query;
    sys.htp.p('Statement = '||l_report.sql_query);
    for i in 1..l_report.binds.count
    loop
        sys.htp.p(i||'. '||l_report.binds(i).name||' = '||l_report.binds(i).value);
    end loop;
END;
```

Example 2

The following example shows how to use the GET_REPORT function to retrieve Group By view query defined in the current report page with region 2505704029884282.

```
DECLARE
   l_report APEX_IR.T_REPORT;
BEGIN
   l_report := APEX_IR.GET_REPORT (
                   p_page_id        => :APP_PAGE_ID,
                   p_region_id      => 2505704029884282,
                   p_view           => APEX_IR.C_VIEW_GROUPBY );
   sys.htp.p( 'Statement = '||l_report.sql_query );
END;
```

12.13 RESET_REPORT Procedure Signature 1

This procedure resets report settings to the developer defined default settings using the report ID.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.RESET_REPORT(
    p_page_id   IN NUMBER,
    p_region_id IN NUMBER,
    p_report_id IN NUMBER DEFAULT NULL);
```

Parameters

Table 12-13 RESET_REPORT Procedure Signature 1 Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_region_id	The interactive report region ID.
p_report_id	The saved report ID within the current application page. If p_report_id is null, it resets the last viewed report settings.

Example

The following example shows how to use the RESET_REPORT procedure signature 1 to reset interactive report settings with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.RESET_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_id    => 880629800374638220);
END;
```

12.14 RESET_REPORT Procedure Signature 2

This procedure resets report settings using the report alias.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.RESET_REPORT(
    p_page_id      IN NUMBER,
    p_region_id    IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 12-14 RESET_REPORT Procedure Signature 2 Parameters

Parameter	Description
`p_page_id`	Page of the current Application Express application that contains an interactive report.
`p_region_id`	The interactive report region ID.
`p_report_alias`	The saved report alias within the current application page. If `p_report_alias` is null, it resets the last viewed report settings.

Example

The following example shows how to use the `RESET_REPORT` procedure to reset interactive report settings with a report alias of `CATEGORY_REPORT` in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.RESET_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_alias => 'CATEGORY_REPORT');
END;
```

13

APEX_ITEM

You can use the `APEX_ITEM` package to create form elements dynamically based on a SQL query instead of creating individual items page by page.

13.1 CHECKBOX2 Function

This function creates check boxes.

Syntax

```
APEX_ITEM.CHECKBOX2(
    p_idx                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT NULL,
    p_attributes                IN    VARCHAR2 DEFAULT NULL,
    p_checked_values            IN    VARCHAR2 DEFAULT NULL,
    p_checked_values_delimiter  IN    VARCHAR2 DEFAULT ':',
    p_item_id                   IN    VARCHAR2 DEFAULT NULL,
    p_item_label                IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-1 CHECKBOX2 Parameters

Parameter	Description
`p_idx`	Number that determines which `APEX_APPLICATION` global variable is used. Valid range of values is 1 to 50. For example 1 creates `F01` and 2 creates `F02`
`p_value`	Value of a check box, hidden field, or input form item
`p_attributes`	Controls the size of the text field
`p_checked_values`	Values to be checked by default
`p_checked_values_delimiter`	Delimits the values in the previous parameter, `p_checked_values`
`p_item_id`	HTML attribute ID for the `<input>` tag
`p_item_label`	Invisible label created for the item

Examples of Default Check Box Behavior

The following example demonstrates how to create a selected check box for each employee in the `emp` table.

```
SELECT APEX_ITEM.CHECKBOX2(1,empno,'CHECKED') "Select",
    ename, job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to have all check boxes for employees display without being selected.

```
SELECT APEX_ITEM.CHECKBOX2(1,empno) "Select",
    ename, job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to select the check boxes for employees who work in department 10.

```
SELECT APEX_ITEM.CHECKBOX2(1,empno,DECODE(deptno,10,'CHECKED',NULL)) "Select",
    ename, job
FROM   emp
ORDER BY 1
```

The next example demonstrates how to select the check boxes for employees who work in department 10 or department 20.

```
SELECT APEX_ITEM.CHECKBOX2(1,deptno,NULL,'10:20',':') "Select",
    ename, job
FROM   emp
ORDER BY 1
```

Creating an On-Submit Process

If you are using check boxes in your application, you might need to create an On Submit process to perform a specific type of action on the selected rows. For example, you could have a Delete button that uses the following logic:

```
SELECT APEX_ITEM.CHECKBOX2(1,empno) "Select",
    ename, job
FROM   emp
ORDER  by 1
```

Consider the following sample on-submit process:

```
FOR I in 1..APEX_APPLICATION.G_F01.COUNT LOOP
    DELETE FROM emp WHERE empno = to_number(APEX_APPLICATION.G_F01(i));
END LOOP;
```

The following example demonstrates how to create unselected checkboxes for each employee in the emp table, with a unique ID. This is useful for referencing records from within JavaScript code:

```
SELECT APEX_ITEM.CHECKBOX2(1,empno,NULL,NULL,NULL,'f01_#ROWNUM#') "Select",
    ename, job
FROM   emp
ORDER BY 1
```

13.2 DATE_POPUP Function

Use this function with forms that include date fields. The DATE_POPUP function dynamically generates a date field that has a popup calendar button.

Syntax

```
APEX_ITEM.DATE_POPUP(
    p_idx                       IN    NUMBER,
    p_row                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT NULL,
    p_date_format               IN    DATE DEFAULT 'DD-MON-YYYY',
    p_size                      IN    NUMBER DEFAULT 20,
    p_maxlength                 IN    NUMBER DEFAULT 2000,
    p_attributes                IN    VARCHAR2 DEFAULT NULL,
    p_item_id                   IN    VARCHAR2 DEFAULT NULL,
    p_item_label                IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-2 DATE_POPUP Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used.Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02
p_row	This parameter is deprecated. Anything specified for this value is ignored
p_value	Value of a field item
p_date_format	Valid database date format
p_size	Controls HTML tag attributes (such as disabled)
p_maxlength	Determines the maximum number of enterable characters. Becomes the maxlength attribute of the <input> HTML tag
p_attributes	Extra HTML parameters you want to add
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Example

The following example demonstrates how to use APEX_ITEM.DATE_POPUP to create popup calendar buttons for the hiredate column.

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno)||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hd,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

See Also:

Oracle Database SQL Language Reference for information about the TO_CHAR or TO_DATE functions

13.3 DATE_POPUP2 Function

Use this function with forms that include date fields. The DATE_POPUP2 function dynamically generates a date field that has a jQuery based popup calendar with button.

Syntax

```
APEX_ITEM.DATE_POPUP2(
    p_idx                  in number,
    p_value                in date     default null,
    p_date_format          in varchar2 default null,
    p_size                 in number   default 20,
    p_maxLength            in number   default 2000,
    p_attributes           in varchar2 default null,
    p_item_id              in varchar2 default null,
    p_item_label           in varchar2 default null,
    p_default_value        in varchar2 default null,
    p_max_value            in varchar2 default null,
    p_min_value            in varchar2 default null,
    p_show_on              in varchar2 default 'button',
    p_number_of_months     in varchar2 default null,
    p_navigation_list_for  in varchar2 default 'NONE',
    p_year_range           in varchar2 default null,
    p_validation_date      in varchar2 default null)
    RETURN VARCHAR2;
```

Parameters

Table 13-3 DATE_POPUP2 Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used.Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02.
p_value	Value of a field item
p_date_format	Valid database date format
p_size	Controls HTML tag attributes (such as disabled)
p_maxlength	Determines the maximum number of enterable characters. Becomes the maxlength attribute of the <input> HTML tag
p_attributes	Extra HTML parameters you want to add
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item
p_default_value	The default date which should be selected in DatePicker calendar popup
p_max_value	The Maximum date that can be selected from the datepicker
p_min_value	The Minimum date that can be selected from the datepicker.
p_show_on	Determines when the datepicker displays, on button click or on focus of the item or both.

Table 13-3 (Cont.) DATE_POPUP2 Parameters

Parameter	Description
p_number_of_months	Determines number of months displayed. Value should be in array formats follows: [row,column]
p_navigation_list_for	Determines if a select list is displayed for Changing Month, Year or Both. Possible values include: MONTH,YEAR,MONTH_AND_YEAR and default is null.
p_year_range	The range of years displayed in the year selection list.
p_validation_date	Used to store the Date value for the which date validation failed

See Also:

Oracle Database SQL Language Reference for information about the `TO_CHAR` or `TO_DATE` functions

13.4 DISPLAY_AND_SAVE Function

Use this function to display an item as text, but save its value to session state.

Syntax

```
APEX_ITEM.DISPLAY_AND_SAVE(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-4 DISPLAY_AND_SAVE Parameters

Parameter	Description
`p_idx`	Number that determines which `APEX_APPLICATION` global variable is used.Valid range of values is 1 to 50. For example, 1 creates `F01` and 2 creates `F02`
`p_value`	Current value
`p_item_id`	HTML attribute ID for the `<span>` tag
`p_item_label`	Invisible label created for the item

Example

The following example demonstrates how to use the APEX_ITEM.DISPLAY_AND_SAVE function.

```
SELECT APEX_ITEM.DISPLAY_AND_SAVE(10,empno) c FROM emp
```

13.5 HIDDEN Function

This function dynamically generates hidden form items.

Syntax

```
APEX_ITEM.HIDDEN(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL
) RETURN VARCHAR2;
```

Parameters

Table 13-5 HIDDEN Parameters

Parameter	Description
p_idx	Number to identify the item you want to generate. The number determines which G_FXX global is populated **See Also:** " APEX_APPLICATION (page 1-1)"
p_value	Value of the hidden input form item
p_attributes	Extra HTML parameters you want to add
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Example

Typically, the primary key of a table is stored as a hidden column and used for subsequent update processing, for example:

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno)||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

The previous query could use the following page process to process the results:

```
BEGIN
    FOR i IN 1..APEX_APPLICATION.G_F01.COUNT LOOP
        UPDATE emp
            SET
                ename=APEX_APPLICATION.G_F02(i),
                job=APEX_APPLICATION.G_F03(i),
                    hiredate=to_date(APEX_APPLICATION.G_F04(i),'dd-mon-yyyy'),
                    sal=APEX_APPLICATION.G_F05(i),
                    comm=APEX_APPLICATION.G_F06(i)
        WHERE empno=to_number(APEX_APPLICATION.G_F01(i));
    END LOOP;
END;
```

Note that the `G_F01` column (which corresponds to the hidden `EMPNO`) is used as the key to update each row.

13.6 MD5_CHECKSUM Function

Use this function for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces hidden form field(s) with a name attribute equal to 'fcs' and as value a MD5 checksum based on up to 50 inputs. APEX_ITEM.MD5_CHECKSUM also produces an MD5 checksum using Oracle database DBMS_CRYPTO:

```
UTL_RAW.CAST_TO_RAW(DBMS_CRYPTO.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network.

Syntax

```
APEX_ITEM.MD5_CHECKSUM(
    p_value01   IN    VARCHAR2 DEFAULT NULL,
    p_value02   IN    VARCHAR2 DEFAULT NULL,
    p_value03   IN    VARCHAR2 DEFAULT NULL,
    ...
    p_value50   IN    VARCHAR2 DEFAULT NULL,
    p_col_sep   IN    VARCHAR2 DEFAULT '|',
    p_item_id   IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-6 MD5_CHECKSUM Parameters

Parameter	Description
`p_value01` ... `p_value50`	Fifty available inputs. If no parameters are supplied, the default to NULL
`p_col_sep`	String used to separate `p_value` inputs. Defaults to the pipe symbol (`|`)
`p_item_id`	ID of the HTML form item

Example

This function generates hidden form elements with the name 'fcs'. The values can subsequently be accessed by using the APEX_APPLICATION.G_FCS array.

```
SELECT APEX_ITEM.MD5_CHECKSUM(ename,job,sal) md5_cks,
       ename, job, sal
FROM emp
```

13.7 MD5_HIDDEN Function

Use this function for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces a hidden form field with a MD5 checksum as value which is based on up to 50 inputs. APEX_ITEM.MD5_HIDDEN also produces an MD5 checksum using Oracle database DBMS_CRYPTO:

```
UTL_RAW.CAST_TO_RAW(DBMS_CRYPTO.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network

Syntax

```
APEX_ITEM.MD5_HIDDEN(
    p_idx       IN    NUMBER,
    p_value01   IN    VARCHAR2 DEFAULT NULL,
    p_value02   IN    VARCHAR2 DEFAULT NULL,
    p_value03   IN    VARCHAR2 DEFAULT NULL,
    ...
    p_value50   IN    VARCHAR2 DEFAULT NULL,
    p_col_sep   IN    VARCHAR2 DEFAULT '|',
    p_item_id   IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-7 MD5_HIDDEN Parameters

Parameter	Description
p_idx	Indicates the form element to be generated. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column
p_value01 ... p_value50	Fifty available inputs. Parameters not supplied default to NULL
p_col_sep	String used to separate p_value inputs. Defaults to the pipe symbol (|)
p_item_id	ID of the HTML form item

Example

The p_idx parameter specifies the FXX form element to be generated. In the following example, 7 generates F07. Also note that an HTML hidden form element is generated.

```
SELECT APEX_ITEM.MD5_HIDDEN(7,ename,job,sal)md5_h, ename, job, sal
FROM emp
```

13.8 POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application shared list of values (LOV). Like other available functions in the `APEX_ITEM` package, `POPUP_FROM_LOV` function is designed to generate forms with `F01` to `F50` form array elements.

Syntax

```
APEX_ITEM.POPUP_FROM_LOV(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-8 POPUP_FROM_LOV Parameters

Parameter	Description
`p_idx`	Form element name. For example, `1` equals `F01` and `2` equals `F02`. Typically, `p_idx` is a constant for a given column
`p_value`	Form element current value. This value should be one of the values in the `p_lov_name` parameter
`p_lov_name`	Named LOV used for this popup
`p_width`	Width of the text box
`p_max_length`	Maximum number of characters that can be entered in the text box
`p_form_index`	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the `#FORM_OPEN#` substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 13-8 (Cont.) POPUP_FROM_LOV Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent: • <lt; for < • <gt; for > • <amp; for & Range of values is YES and NO. If YES, special characters are escaped. This parameter is useful if you know your query returns illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup from an LOV named DEPT_LOV.

```
SELECT APEX_ITEM.POPUP_FROM_LOV (1,deptno,'DEPT_LOV') dt
FROM emp
```

13.9 POPUP_FROM_QUERY Function

This function generates an HTML popup select list from a query. Like other available functions in the APEX_ITEM package, the POPUP_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUP_FROM_QUERY(

    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-9 POPUP_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_query parameter.
p_lov_query	SQL query that is expected to select two columns (a display column and a return column). For example: `SELECT dname, deptno FROM dept`
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. • < for < • > for > • & for & Range of values is YES and NO. If YES, special characters are escaped. This parameter is useful if you know your query returns illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup select list from the emp table.

```
SELECT APEX_ITEM.POPUP_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

13.10 POPUPKEY_FROM_LOV Function

This function generates a popup key select list from a shared list of values (LOV). Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_LOV function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUPKEY_FROM_LOV(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Although the text field associated with the popup displays in the first column in the LOV query, the actual value is specified in the second column in the query.

Parameters

Table 13-10 POPUPKEY_FROM_LOV Parameters

Parameter	Description
p_idx	Identifies a form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: `SELECT APEX_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt, APEX_ITEM.HIDDEN(3,empno) eno`
p_value	Indicates the current value. This value should be one of the values in the P_LOV_NAME parameter.
p_lov_name	Identifies a named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.

Table 13-10 (Cont.) POPUPKEY_FROM_LOV Parameters

Parameter	Description
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. • < for < • > for > • & for & This parameter is useful if you know your query returns illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Example

The following example demonstrates how to generate a popup key select list from a shared list of values (LOV).

```
SELECT APEX_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt
FROM emp
```

13.11 POPUPKEY_FROM_QUERY Function

This function generates a popup key select list from a SQL query. Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUPKEY_FROM_QUERY(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
```

```
p_lov_query        IN    VARCHAR2,
p_width            IN    VARCHAR2 DEFAULT NULL,
p_max_length       IN    VARCHAR2 DEFAULT NULL,
p_form_index       IN    VARCHAR2 DEFAULT '0',
p_escape_html      IN    VARCHAR2 DEFAULT NULL,
p_max_elements     IN    VARCHAR2 DEFAULT NULL,
p_attributes       IN    VARCHAR2 DEFAULT NULL,
p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
p_item_id          IN    VARCHAR2 DEFAULT NULL,
p_item_label       IN    VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 13-11 POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column. Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: `SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt, APEX_ITEM.HIDDEN(3,empno) eno`
p_value	Form element current value. This value should be one of the values in the P_LOV_QUERY parameter.
p_lov_query	LOV query used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. • < for < • > for > • & for & This parameter is useful if you know your query returns illegal HTML.

Table 13-11 (Cont.) POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to generate a popup select list from a SQL query.

```
SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

13.12 RADIOGROUP Function

This function generates a radio group from a SQL query.

Syntax

```
APEX_ITEM.RADIOGROUP(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_selected_value   IN    VARCHAR2 DEFAULT NULL,
    p_display          IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_onblur           IN    VARCHAR2 DEFAULT NULL,
    p_onchange         IN    VARCHAR2 DEFAULT NULL,
    p_onfocus          IN    VARCHAR2 DEFAULT NULL,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-12 RADIOGROUP Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50.For example 1 creates F01 and 2 creates F02.
p_value	Value of the radio group.

Table 13-12 (Cont.) RADIOGROUP Parameters

Parameter	Description
p_selected_value	Value that should be selected.
p_display	Text to display next to the radio option.
p_attributes	Extra HTML parameters you want to add.
p_onblur	JavaScript to execute in the onBlur event.
p_onchange	JavaScript to execute in the onChange event.
p_onfocus	JavaScript to execute in the onFocus event.
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Example

The following example demonstrates how to select department 20 from the emp table as a default in a radio group.

```
SELECT APEX_ITEM.RADIOGROUP (1,deptno,'20',dname) dt
FROM   dept
ORDER  BY 1
```

13.13 SELECT_LIST Function

This function dynamically generates a static select list. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT NULL,
    p_list_values   IN   VARCHAR2 DEFAULT NULL,
    p_attributes    IN   VARCHAR2 DEFAULT NULL,
    p_show_null     IN   VARCHAR2 DEFAULT 'NO',
    p_null_value    IN   VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN   VARCHAR2 DEFAULT '%',
    p_item_id       IN   VARCHAR2 DEFAULT NULL,
    p_item_label    IN   VARCHAR2 DEFAULT NULL,
    p_show_extra    IN   VARCHAR2 DEFAULT 'YES')
    RETURN VARCHAR2;
```

Parameters

Table 13-13 SELECT_LIST Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.
p_value	Current value. This value should be a value in the P_LIST_VALUES parameter.
p_list_values	List of static values separated by commas. Displays values and returns values that are separated by semicolons. Note that this is only available in the SELECT_LIST function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a static select list that displays Yes, returns Y, defaults to Y, and generates a F01 form item.

```
SELECT APEX_ITEM.SELECT_LIST(1,'Y','Yes;Y,No;N')yn
FROM emp
```

The following example demonstrates the use of APEX_ITEM.SELECT_LIST to generate a static select list where:

- A form array element F03 is generated (p_idx parameter).
- The initial value for each element is equal to the value for deptno for the row from emp (p_value parameter).
- The select list contains 4 options (p_list_values parameter).
- The text within the select list displays in red (p_attributes parameter).
- A null option is displayed (p_show_null) and this option displays -Select- as the text (p_null_text parameter).

- An HTML ID attribute is generated for each row, where #ROWNUM# is substituted for the current row rownum (p_item_id parameter). (So an ID of 'f03_4' is generated for row 4.)
- A HTML label element is generated for each row (p_item_label parameter).
- The current value for deptno is displayed, even if it is not contained with the list of values passed in the p_list_values parameter (p_show_extra parameter).

```
SELECT  empno "Employee #",
    ename "Name",
    APEX_ITEM.SELECT_LIST(
        p_idx           =>   3,
        p_value         =>   deptno,
        p_list_values   =>   'ACCOUNTING;10,RESEARCH;20,SALES;30,OPERATIONS;40',
        p_attributes    =>   'style="color:red;"',
        p_show_null     =>   'YES',
        p_null_value    =>   NULL,
        p_null_text     =>   '-Select-',
        p_item_id       =>   'f03_#ROWNUM#',
        p_item_label    =>   'Label for f03_#ROWNUM#',
        p_show_extra    =>   'YES') "Department"
  FROM  emp;
```

13.14 SELECT_LIST_FROM_LOV Function

This function dynamically generates select lists from a shared list of values (LOV). Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements. This function is the same as SELECT_LIST_FROM_LOV, but its return value is CLOB. Use this function in SQL queries where you need to handle a column value longer than 4000 characters.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT NULL,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT NULL,
    p_show_null     IN   VARCHAR2 DEFAULT 'YES',
    p_null_value    IN   VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN   VARCHAR2 DEFAULT '%',
    p_item_id       IN   VARCHAR2 DEFAULT NULL,
    p_item_label    IN   VARCHAR2 DEFAULT NULL,
    p_show_extra    IN   VARCHAR2 DEFAULT 'YES')
    RETURN VARCHAR2;
```

Parameters

Table 13-14 SELECT_LIST_FROM_LOV Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_lov parameter.

Table 13-14 (Cont.) SELECT_LIST_FROM_LOV Parameters

Parameter	Description
p_lov	Text name of an application list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV(2,job,'JOB_FLOW_LOV')job
FROM emp
```

13.15 SELECT_LIST_FROM_LOV_XL Function

This function dynamically generates very large select lists (greater than 32K) from a shared list of values (LOV). Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements. This function is the same as SELECT_LIST_FROM_LOV, but its return value is CLOB. Returned values will be limited to 32k.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV_XL(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT NULL,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT NULL,
    p_show_null     IN   VARCHAR2 DEFAULT 'YES',
    p_null_value    IN   VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN   VARCHAR2 DEFAULT '%',
    p_item_id       IN   VARCHAR2 DEFAULT NULL,
    p_item_label    IN   VARCHAR2 DEFAULT NULL,
    p_show_extra    IN   VARCHAR2 DEFAULT 'YES')
    RETURN CLOB;
```

Parameters

Table 13-15 SELECT_LIST_FROM_LOV_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_lov parameter.
p_lov	Text name of a list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates how to create a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV_XL(2,job,'JOB_FLOW_LOV')job
FROM emp
```

13.16 SELECT_LIST_FROM_QUERY Function

This function dynamically generates a select list from a query. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_QUERY(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT NULL,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT NULL,
    p_show_null     IN    VARCHAR2 DEFAULT 'YES',
    p_null_value    IN    VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN    VARCHAR2 DEFAULT '%',
    p_item_id       IN    VARCHAR2 DEFAULT NULL,
```

```
p_item_label    IN    VARCHAR2 DEFAULT NULL,
p_show_extra    IN    VARCHAR2 DEFAULT 'YES')
RETURN VARCHAR2;
```

Parameters

Table 13-16 SELECT_LIST_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_query parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY function. Also note, if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY(3,job,'SELECT DISTINCT job FROM emp')job
FROM emp
```

13.17 SELECT_LIST_FROM_QUERY_XL Function

This function is the same as SELECT_LIST_FROM_QUERY, but its return value is a CLOB. This allows its use in SQL queries where you need to handle a column value longer than 4000 characters. Returned values will be limited to 32K. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT NULL,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT NULL,
    p_show_null     IN    VARCHAR2 DEFAULT 'YES',
    p_null_value    IN    VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN    VARCHAR2 DEFAULT '%',
    p_item_id       IN    VARCHAR2 DEFAULT NULL,
    p_item_label    IN    VARCHAR2 DEFAULT NULL,
    p_show_extra    IN    VARCHAR2 DEFAULT 'YES')
    RETURN CLOB;
```

Parameters

Table 13-17 SELECT_LIST_FROM_QUERY_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_query parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY_XL function. Also note, if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(3,job,'SELECT DISTINCT job FROM emp')job
FROM emp
```

13.18 SWITCH Function

This function dynamically generates flip toggle item. If On/Off value and label are not passed, it renders Yes/No toggle. Similar to other functions available in the APEX_ITEM package, switch function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SWITCH(
     p_idx IN NUMBER,
     p_value IN VARCHAR2,
     p_on_value IN VARCHAR2 DEFAULT 'Y',
     p_on_label IN VARCHAR2 DEFAULT 'Yes',
     p_off_value IN VARCHAR2 DEFAULT 'N',
     p_off_label IN VARCHAR2 DEFAULT 'No',
     p_item_id IN VARCHAR2 DEFAULT NULL,
     p_item_label IN VARCHAR2,
     p_attributes IN VARCHAR2 DEFAULT NULL)
     RETURN VARCHAR2;
```

Parameters

Table 13-18 SWITCH Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.
p_value	Form element current value.
p_on_value	The value of the item if the user picks **On** option.
p_on_label	The display text for the **On** option.
p_off_value	The value of the item if the user picks **Off** option.
p_off_label	The display text for the **Off** option.
p_item_id	HTML attribute ID for the <input> tag. Try concatenating some string with rownum to make it unique.
p_item_label	Invisible label created for the item.
p_attributes	Additional HTML attributes to use for the form item.

Example

The following example demonstrates the use of APEX_ITEM.SWITCH to generate a Yes/No flip toggle item where:

- A form array element F01 will be generated (p_idx parameter).

- The initial value for each element will be equal to N (`p_value` parameter).
- A HTML ID attribute will be generated for each row with the current rownum to uniquely identify. (`p_item_id` parameter). An ID of `'IS_MANAGER_2'` is generated for row 2.)
- A HTML label element will be generated for each row (`p_item_label` parameter).

```
SELECT
  ename "Name",
  APEX_ITEM.SWITCH (
       p_idx => 1,
       p_value => 'N',
   p_item_id => 'IS_MANAGER_'||rownum,
       p_item_label => apex_escape.html(ename)||': Is Manager' )
"Is Manager"
FROM emp;
```

13.19 TEXT Function

This function generates text fields (or text input form items) from a SQL query.

Syntax

```
APEX_ITEM.TEXT(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_size        IN    NUMBER DEFAULT NULL,
    p_maxlength   IN    NUMBER DEFAULT NULL,
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-19 TEXT Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number determines which `G_FXX` global is populated. **See Also:** " APEX_APPLICATION (page 1-1)"
`p_value`	Value of a text field item.
`p_size`	Controls HTML tag attributes (such as disabled).
`p_maxlength`	Maximum number of characters that can be entered in the text box.
`p_attributes`	Extra HTML parameters you want to add.
`p_item_id`	HTML attribute ID for the `<input>` tag.
`p_item_label`	Invisible label created for the item.

Example

The following sample query demonstrates how to generate one update field for each row. Note that the `ename`, `sal`, and `comm` columns use the `APEX_ITEM.TEXT` function to generate an HTML text field for each row. Also, notice that each item in the query is passed a unique `p_idx` parameter to ensure that each column is stored in its own array.

```
SELECT
  empno,
  APEX_ITEM.HIDDEN(1,empno)||
  APEX_ITEM.TEXT(2,ename) ename,
  APEX_ITEM.TEXT(3,job) job,
  mgr,
  APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
  APEX_ITEM.TEXT(5,sal) sal,
  APEX_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

13.20 TEXTAREA Function

This function creates text areas.

Syntax

```
APEX_ITEM.TEXTAREA(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_rows        IN    NUMBER DEAULT 40,
    p_cols        IN    NUMBER DEFAULT 4,
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 13-20 TEXTAREA Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number determines which `G_FXX` global is populated. **See Also:** " APEX_APPLICATION (page 1-1)"
`p_value`	Value of the text area item.
`p_rows`	Height of the text area (HTML rows attribute)
`p_cols`	Width of the text area (HTML column attribute).
`p_attributes`	Extra HTML parameters you want to add.
`p_item_id`	HTML attribute ID for the `<textarea>` tag.

Table 13-20 (Cont.) TEXTAREA Parameters

Parameter	Description
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to create a text area based on a SQL query.

```
SELECT APEX_ITEM.TEXTAREA(3,ename,5,80) a
FROM emp
```

13.21 TEXT_FROM_LOV Function

Use this function to display an item as text, deriving the display value of the named LOV.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_lov         IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 13-21 TEXT_FROM_LOV Parameters

Parameter	Description
p_value	Value of a field item. Note that if p_value is not located in the list of values, p_null_text is value displayed.
p_lov	Text name of a shared list of values. This list of values must be defined in your application.
p_null_text	Value displayed when the value of the field item is NULL.

Example

The following example demonstrates how to derive the display value from a named LOV (EMPNO_ENAME_LOV).

```
SELECT APEX_ITEM.TEXT_FROM_LOV(empno,'EMPNO_ENAME_LOV') c FROM emp
```

13.22 TEXT_FROM_LOV_QUERY Function

Use this function to display an item as text, deriving the display value from a list of values query.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV_QUERY (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_query       IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 13-22 TEXT_FROM_LOV_QUERY Parameters

Parameter	Description
p_value	Value of a field item.
p_query	SQL query that is expected to select two columns, a display column and a return column. For example: `SELECT dname, deptno FROM dept` Note if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.
p_null_text	Value to be displayed when the value of the field item is NULL or a corresponding entry is not located for the value p_value in the list of values query.

Example

The following example demonstrates how to derive the display value from a query.

```
SELECT APEX_ITEM.TEXT_FROM_LOV_QUERY(empno,'SELECT ename, empno FROM emp') c from emp
```

14

APEX_JAVASCRIPT

The APEX_JAVASCRIPT package provides utility functions for adding dynamic JavaScript code to HTTP output. This package is usually used for plug-in development.

14.1 ADD_3RD_PARTY_LIBRARY_FILE Procedure

This procedure adds the script tag to load a 3rd party javascript library file and also takes into account the specified Content Delivery Network for the application. Supported libraries include: jQuery, jQueryUI, and jQuery Mobile.

Syntax

```
add_3rd_party_library_file (
    p_library in varchar2,
    p_file_name in varchar2,
    p_directory in varchar2 default null,
    p_version in varchar2 default null );
```

Parameters

Table 14-1 ADD_3RD_PARTY_LIBRARY_FILE Parameters

Parameters	Description
p_library	Use one of the c_library_* constants
p_file_name	Specifies the file name without version, .min and .js
p_directory	Directory where the file p_file_name is located (optional)
p_version	If no value is provided then the same version Application Express ships is used (optional)

Example

This example loads the JavaScript file of the Draggable feature of jQuery UI.

```
apex_javascript.add_3rd_party_library_file (
     p_library   =>apex_javascript.c_library_jquery_ui,
     p_file_name => 'jquery.ui.draggable' )
```

14.2 ADD_ATTRIBUTE Function Signature 1

This function returns the attribute and the attribute's escaped text surrounded by double quotation marks.

Note:

This function does not escape HTML tags. It only prevents HTML tags from breaking the JavaScript object attribute assignment. To prevent XSS (cross site scripting) attacks, you must also call SYS.HTF.ESCAPE_SC to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name       IN VARCHAR2,
    p_value      IN VARCHAR2,
    p_omit_null  IN BOOLEAN:=TRUE,
    p_add_comma  IN BOOLEAN:=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-2 ADD_ATTRIBUTE Signature 1 Parameters

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Text to be assigned to the JavaScript object attribute.
p_omit_null	If set to TRUE and p_value is empty, returns NULL.

Table 14-2 (Cont.) ADD_ATTRIBUTE Signature 1 Parameters

Parameter	Description
p_add_comma	If set to TRUE, a trailing comma is added when a value is returned.

Example

Adds a call to the addEmployee JavaScript function and passes in a JavaScript object with different attribute values. The output of this call looks like:

```
addEmployee(
  {"FirstName":"John",
   "LastName":"Doe",
   "Salary":2531.29,
   "Birthday":new Date(1970,1,15,0,0,0),
   "isSalesman":true
  });
```

As the last attribute you should use the parameter combination FALSE (p_omit_null), FALSE (p_add_comma) so that the last attribute is always generated. This avoids that you have to check for the other parameters if a trailing comma should be added or not.

```
apex_javascript.add_onload_code (
    'addEmployee('||
        '{'||
        apex_javascript.add_attribute('FirstName',
sys.htf.escape_sc(l_first_name))||
        apex_javascript.add_attribute('LastName',   sys.htf.escape_sc(l_last_name))||
        apex_javascript.add_attribute('Salary',     l_salary)||
        apex_javascript.add_attribute('Birthday',   l_birthday)||
        apex_javascript.add_attribute('isSalesman', l_is_salesman, false, false)||
        '});' );
```

14.3 ADD_ATTRIBUTE Function Signature 2

This function returns the attribute and the attribute's number.

Syntax

```
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name       IN VARCHAR2,
    p_value      IN NUMBER,
    p_omit_null  IN BOOLEAN:=TRUE,
    p_add_comma  IN BOOLEAN:=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-3 ADD_ATTRIBUTE Signature 2 Parameters

Parameter	Description
p_name	Name of the JavaScript object attribute.

Table 14-3 (Cont.) ADD_ATTRIBUTE Signature 2 Parameters

Parameter	Description
p_value	Number which should be assigned to the JavaScript object attribute.
p_omit_null	If set to TRUE and p_value is empty, returns NULL.
p_add_comma	If set to TRUE, a trailing comma is added when a value is returned.

Example

See example for ADD_ATTRIBUTE Function Signature 1 (page 14-2).

14.4 ADD_ATTRIBUTE Function Signature 3

This function returns the attribute and a JavaScript boolean of TRUE, FALSE, or NULL.

Syntax

```
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name       IN VARCHAR2,
    p_value      IN BOLLEAN,
    p_omit_null  IN BOOLEAN:=TRUE,
    p_add_comma  IN BOOLEAN:=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-4 ADD_ATTRIBUTE Signature 3 Parameters

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Boolean assigned to the JavaScript object attribute.
p_omit_null	If p_omit_null is TRUE and p_value is NULL the function returns NULL.
p_add_comma	If set to TRUE a trailing comma is added when a value is returned.

Example

See example for ADD_ATTRIBUTE Function Signature 1 (page 14-2)

14.5 ADD_ATTRIBUTE Function Signature 4

This function returns the attribute and the attribute's date. If p_value is null the value null is returned.

Syntax

```
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name       IN VARCHAR2,
    p_value      IN DATE,
    p_omit_null  IN BOOLEAN:=TRUE,
    p_add_comma  IN BOOLEAN:=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-5 ADD_ATTRIBUTE SIgnature 4 Parameters

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Date assigned to the JavaScript object attribute.
p_omit_null	If p_omit_null is TRUE and p_value is NULL the function returns NULL.
p_add_comma	If set to TRUE a trailing comma is added when a value is returned.

Example

See example for ADD_ATTRIBUTE Function Signature 1 (page 14-2)

14.6 ADD_INLINE_CODE Procedure

This procedure adds a code snippet that is included inline into the HTML output. For example, you can use this procedure to add new functions or global variable declarations.

Note:

If you want to execute code you should use ADD_ONLOAD_CODE Procedure (page 14-8).

Syntax

```
APEX_JAVASCRIPT.ADD_INLINE_CODE (
    p_code       IN VARCHAR2,
    p_key        IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 14-6 ADD_INLINE_CODE Parameters

Parameter	Description
p_code	JavaScript code snippet. For example: $s('P1_TEST',123);

Table 14-6 (Cont.) ADD_INLINE_CODE Parameters

Parameter	Description
p_key	Identifier for the code snippet. If specified and a code snippet with the same name has already been added, the new code snippet is ignored. If p_key is NULL the snippet is always added.

Example

The following example includes the JavaScript function `initMySuperWidget` in the HTML output. If the plug-in is used multiple times on the page and the `add_inline_code` is called multiple times, it is added once to the HTML output because all calls have the same value for `p_key`.

```
apex_javascript.add_inline_code (
    p_code => 'function initMySuperWidget(){'||chr(10)||
              '  // do something'||chr(10)||
              '};',
    p_key  => 'my_super_widget_function' );
```

14.7 ADD_LIBRARY Procedure

This procedure adds the script tag to load a JavaScript library. If a library has been added, it is not added a second time.

Syntax

```
APEX_JAVASCRIPT.ADD_LIBRARY (
    p_name                    IN VARCHAR2,
    p_directory               IN VARCHAR2,
    p_version                 IN VARCHAR2 DEFAULT NULL,
    p_check_to_add_minified   IN BOOLEAN DEFAULT FALSE,
    p_skip_extension          IN BOOLEAN  DEFAULT FALSE,
    p_ie_condition            IN VARCHAR2 DEFAULT NULL,
    p_requirejs_module        IN VARCHAR2 DEFAULT NULL,
    p_requirejs_js_expression IN VARCHAR2 DEFAULT NULL,
    p_requirejs_required      IN BOOLEAN DEFAULT FALSE,
    p_key                     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 14-7 ADD_LIBRARY Parameters

Parameter	Description
p_name	Name of the JavaScript file. Must not use `.js` when specifying.
p_directory	Directory where JavaScript library is loaded. Must have a trailing slash.
p_version	Version identifier.

Table 14-7 (Cont.) ADD_LIBRARY Parameters

Parameter	Description
p_check_to_add_minified	If TRUE, the procedure tests if it is appropriate to add .min extension and add it if appropriate. This is added if an application is not running in DEBUG mode, and omitted when in DEBUG mode.
p_skip_extension	If TRUE the extension .js is NOT added.
p_ie_condition	Condition which is used as Internet Explorer condition.
p_requirejs_module	Module name which is used to expose the library to RequireJS.
p_requirejs_js_expression	JavaScript expression which is used to expose the library to the RequireJS module.
p_requirejs_required	This has to be true if the library uses RequireJS in its code to loading other JavaScript files.
p_key	Name used to indicate if the library has already been loaded. If not specified, defaults to p_directory||p_name||p_version.

Example

The following example includes the JavaScript library file named hammer-2.0.4.min.js (if the application has not been started from the Builder), or hammer-2.0.4.js (if the application has been started from the Builder or is running in DEBUG mode), from the directory specified by p_plugin.file_prefix. Since p_skip_extension is not specified, this defaults to .js. Also, since p_key is not specified, the key defaults to p_plugin.file_prefix||hammer-2.0.4. Hammer is a JavaScript library which exposes itself to RequireJS using hammerjs as module name.

```
apex_javascript.add_library (
    p_name                   => 'hammer-2.0.4#MIN#',
    p_directory              => p_plugin.file_prefix,
    p_requirejs_module       => 'hammerjs',
    p_requirejs_js_expression => 'Hammer' );
```

14.8 ADD_REQUIREJS Procedure

This procedure adds the script tag to load the RequireJS library.

Syntax

```
procedure add_requirejs;
```

14.9 ADD_REQUIREJS_DEFINE Procedure

This procedure adds a RequireJS define after RequireJS has been loaded to let it know about the existence of a library.

Syntax

```
APEX_JAVASCRIPT.add_requirejs_define (
          p_module        in varchar2,
          p_js_expression in varchar2 );
```

Parameters

Table 14-8 ADD_REQUIREJS_DEFINE Parameters

Parameter	Description
p_module	
p_js_expression	

Example

```
apex_javascript.add_requirejs_define (
    p_module        => 'hammerjs',
    p_js_expression => 'Hammer' );
```

14.10 ADD_ONLOAD_CODE Procedure

This procedure adds a javascript code snippet to the HTML output which is executed by the onload event. If an entry with the same key exists it is ignored. If p_key is NULL the snippet is always added.

Syntax

```
APEX_JAVASCRIPT.ADD_ONLOAD_CODE (
    p_code           IN VARCHAR2,
    p_key            IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 14-9 ADD_ONLOAD_CODE Parameters

Parameter	Description
p_code	Javascript code snippet to be executed during the onload event.
p_key	Any name to identify the specified code snippet. If specified, the code snippet is added if there has been no other call with the same p_key. If p_key is NULL the code snippet is always added.

Example

Adds the JavaScript call initMySuperWidget()to the onload buffer. If the plug-in is used multiple times on the page and the add_onload_code is called multiple times, it is added once to the HTML output because all calls have the same value for p_key

```
apex_javascript.add_onload_code (
    p_code => 'initMySuperWidget();'
    p_key  => 'my_super_widget' );
```

14.11 ADD_VALUE Function Signature 1

This function returns the escaped text surrounded by double quotation marks. For example, this string could be returned `"That\'s a test"`.

> **Note:**
>
> This function does not escape HTML tags. It only prevents HTML tags from breaking the JavaScript object attribute assignment. To prevent XSS (cross site scripting) attacks, you must also call `SYS.HTF.ESCAPE_SC` to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN VARCHAR2,
    p_add_comma      IN BOOLEAN :=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-10 ADD_VALUE Signature 1 Parameters

Parameter	Description
`p_value`	Text to be escaped and wrapped by double quotation marks.
`p_add_comma`	If `p_add_comma` is TRUE a trailing comma is added.

Example

This example adds some JavaScript code to the onload buffer. The value of `p_item.attribute_01` is first escaped with `htf.escape_sc` to prevent XSS attacks and then assigned to the JavaScript variable `lTest` by calling `apex_javascript.add_value`. `Add_value` takes care of properly escaping the value and wrapping it with double quotation marks. Because commas are not wanted, `p_add_comma` is set to FALSE.

```
apex_javascript.add_onload_code (
    'var lTest = '||
apex_javascript.add_value(sys.htf.escape_sc(p_item.attribute_01), FALSE)||';'||
chr(10)||
    'showMessage(lTest);' );
```

14.12 ADD_VALUE Function Signature 2

This function returns `p_value` as JavaScript number, if `p_value` is NULL the value null is returned.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN NUMBER,
```

```
    p_add_comma      IN BOOLEAN :=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-11 ADD_VALUE Signature 2 Parameters

Parameter	Description
p_value	Number which should be returned as JavaScript number.
p_add_comma	If p_add_comma is TRUE a trailing comma is added. Default is TRUE.

Example

See example for ADD_VALUE Function Signature 1 (page 14-9) .

14.13 ADD_VALUE Function Signature 3

This function returns p_value as JavaScript boolean. If p_value is NULL the value null is returned.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN BOOLEAN,
    p_add_comma      IN BOOLEAN :=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-12 ADD_VALUE Signature 3 Parameters

Parameter	Description
p_value	Boolean which should be returned as JavaScript boolean.
p_add_comma	If p_add_comma is TRUE a trailing comma is added. Default is TRUE.

Example

See example for ADD_VALUE Function Signature 1 (page 14-9) .

14.14 ADD_VALUE Function Signature 4

This function returns p_value as JavaScript date object, if p_value is NULL the value null is returned.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN NUMBER,
    p_add_comma      IN BOOLEAN :=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 14-13 ADD_VALUE Signature 4 Parameters

Parameter	Description
p_value	Date which should be returned as JavaScript date object.
p_add_comma	If p_add_comma is TRUE a trailing comma is added. Default is TRUE.

Example

See example for ADD_VALUE Function Signature 1 (page 14-9) .

14.15 Escape Function

This function escapes text to be used in JavaScript. This function uses APEX_ESCAPE.JS_LITERAL to escape characters and provide a reference to that other API.

> **Note:**
>
> This function prevents HTML tags from breaking the JavaScript object attribute assignment and also escapes the HTML tags '<' and '>'. It does not escape other HTML tags, therefore to be sure to prevent XSS (cross site scripting) attacks, you must also call SYS.HTF.ESCAPE_SC to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```
APEX_JAVASCRIPT.ESCAPE (
    p_text  IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 14-14 ESCAPE Parameters

Parameter	Description
p_text	Text to be escaped.

Example

Adds some JavaScript code to the onload buffer. The value of p_item.attribute_01 is first escaped with htf.escape_sc to prevent XSS attacks and then escaped with apex_javascript.escape to prevent that special characters like a quotation mark break the JavaScript code.

```
apex_javascript.add_onload_code (
    'var lTest = "'||
apex_javascript.escape(sys.htf.escape_sc(p_item.attribute_01))||'";'||chr(10)||
    'showMessage(lTest);' );
```

15

APEX_JSON

This package includes utilities that parse and generate JSON.

15.1 Package Overview and Examples

To read from a string that contains JSON data, first use `parse()` to convert the string to an internal format. Then use the `get_%` routines (e.g. `get_varchar2()`, `get_number()`, ...) to access the data and `find_paths_like()` to search.

Alternatively, use `to_xmltype()` to convert a JSON string to an xmltype.

This package also contains procedures to generate JSON-formatted output. Use the overloaded `open_%()`, `close_%()` and `write()` procedures for writing.

Example 1

This example parses a JSON string and prints the value of member variable `"a"`.

```
DECLARE
    s varchar2(32767) := '{ "a": 1, "b": ["hello", "world"]}';
BEGIN
    apex_json.parse(s);
    sys.dbms_output.put_line('a is '||apex_json.get_varchar2(p_path => 'a'));
END;
```

Example 2

This example converts a JSON string to XML and uses XMLTABLE to query member values.

```
select col1, col2
from xmltable (
        '/json/row'
        passing apex_json.to_xmltype('[{"col1": 1, "col2": "hello"},'||
                '{"col1": 2, "col2": "world"}]')
        columns
           col1 number path '/row/col1',
           col2 varchar2(5) path '/row/col2' );
```

Example 3

This example writes a nested JSON object to the HTP buffer.

```
BEGIN
    apex_json.open_object;        -- {
    apex_json.write('a', 1);    --   "a":1
    apex_json.open_array('b');  --  ,"b":[
    apex_json.open_object;    --    {
    apex_json.write('c',2); --      "c":2
    apex_json.close_object;   --    }
    apex_json.write('hello'); --   ,"hello"
    apex_json.write('world'); --   ,"world"
    apex_json.close_all;          --  ]
                           -- }
END;
```

15.2 Constants and Data Types

Parser Interface

The following are constants used for the parser interface:

```
subtype t_kind is binary_integer range 1 .. 8;
c_null     constant t_kind := 1;
c_true     constant t_kind := 2;
c_false    constant t_kind := 3;
c_number   constant t_kind := 4;
c_varchar2 constant t_kind := 5;
c_object   constant t_kind := 6;
c_array    constant t_kind := 7;
c_clob     constant t_kind := 8;
```

Storage for JSON Data

JSON data is stored in an index by varchar2 table. The JSON values are stored as records. The discriminator "kind" determines whether the value is null, true, false, a number, a varchar2, a clob, an object or an array. It depends on "kind" which record fields are used and how. If not explicitly mentioned below, the other record fields' values are undefined:

* c_null: -

* c_true: -

* c_false: -

* c_number: number_value contains the number value

* c_varchar2: varchar2_value contains the varchar2 value

* c_clob: clob_value contains the clob

* c_object: object_members contains the names of the object's members

* c_array: number_value contains the array length

```
type t_value is record (
   kind           t_kind,
   number_value   number,
   varchar2_value varchar2(32767),
   clob_value clob,
   object_members apex_t_varchar2 );
type t_values is table of t_value index by varchar2(32767);
```

Default Format for Dates

```
c_date_iso8601 constant varchar2(30) := 'yyyy-mm-dd"T"hh24:mi:ss"Z"';
```

Default JSON Values Table

```
g_values t_values;
```

Errors Thrown for PARSE()

```
e_parse_error     exception;
pragma exception_init(e_parse_error, -20987);
```

15.3 CLOSE_ALL Procedure

This procedure closes all objects and arrays up to the outermost nesting level.

Syntax

```
APEX_JSON.CLOSE_ALL;
```

Parameters

None.

Example

See "Package Overview and Examples (page 15-2)".

15.4 CLOSE_ARRAY Procedure

This procedure writes a close bracket symbol as follows:

```
]
```

Syntax

```
APEX_JSON.CLOSE_ARRAY ();
```

Parameters

None.

Example

See "Package Overview and Examples (page 15-2)".

15.5 CLOSE_OBJECT Procedure

This procedure writes a close curly bracket symbol as follows:

}

Syntax

```
APEX_JSON.CLOSE_OBJECT ();
```

Parameters

None.

Example

See "Package Overview and Examples (page 15-2)".

15.6 DOES_EXIST Function

This function determines whether the given path points to an existing value.

Syntax

```
APEX_JSON.DOES_EXIST (
   p_path             IN VARCHAR2,
   p0                 IN VARCHAR2 DEFAULT NULL,
   p1                 IN VARCHAR2 DEFAULT NULL,
   p2                 IN VARCHAR2 DEFAULT NULL,
   p3                 IN VARCHAR2 DEFAULT NULL,
   p4                 IN VARCHAR2 DEFAULT NULL,
   p_values           IN t_values DEFAULT g_values )
RETURN BOOLEAN;
```

Parameters

Table 15-1 DOES_EXIST Function Parameters

Parameter	Description
`p_path`	Index into `p_values`.
`p[0-4]`	Each %N in `p_path` is replaced by `pN` and every i-th %s or %d is replaced by the `p[i-1]`.
`p_values`	Parsed JSON members. The default is `g_values`.

Returns

Table 15-2 DOES_EXIST Function Returns

Return	Description
`TRUE`	Given path points to an existing value.

Table 15-2 (Cont.) DOES_EXIST Function Returns

Return	Description
FALSE	Given path does not point to an existing value

Example

This example parses a JSON string and prints whether it contains values under a path.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": true } ] }');
    if apex_json.does_exist(p_path => 'items[%d].foo', p0 => 3, p_values =>
j) then
        dbms_output.put_line('found items[3].foo');
    end if;
END;
```

15.7 FIND_PATHS_LIKE Function

This function returns paths into `p_values` that match a given pattern.

Syntax

```
APEX_JSON.FIND_PATHS_LIKE (
    p_return_path      IN VARCHAR2,
    p_subpath          IN VARCHAR2 DEFAULT NULL,
    p_value            IN VARCHAR2 DEFAULT NULL,
    p_values           IN t_values DEFAULT g_values )
RETURN apex_t_varchar2;
```

Parameters

Table 15-3 FIND_PATHS_LIKE Function Parameters

Parameter	Description
`p_return_path`	Search pattern for the return path..
`p_subpath`	Search pattern under `p_return_path` (optional).
`p_value`	Search pattern for value (optional).
`p_values`	Parsed JSON members. The default is `g_values`.

Returns/Raised Errors

Table 15-4 FIND_PATHS_LIKE Function Returns and Raised Errors

Return	Description
`apex_t_varchar2`	Table of paths that match the pattern.
`VALUE_ERROR`	Raises this error if `p_values(p_path)` is not an array or object.

Example

This example parses a JSON string, finds paths that match a pattern, and prints the values under the paths.

```
DECLARE
    j            apex_json.t_values;
    l_paths apex_t_varchar2;
BEGIN
    apex_json.parse(j, '{ "items": [ { "name": "Amulet of Yendor", "magical":
true }, '||
                                     { "name": "Slippers",  "magical": "rather
not" } ]}');
    l_paths := apex_json.find_paths_like (
        p_values         => j,
        p_return_path => 'items[%]',
        p_subpath       => '.magical',
        p_value           => 'true' );
    dbms_output.put_line('Magical items:');
    for i in 1 .. l_paths.count loop
        dbms_output.put_line(apex_json.get_varchar2(p_values => j, p_path =>
l_paths(i)||'.name'));
    end loop;
END;
```

15.8 FREE_OUTPUT Procedure

Frees output resources. Call this procedure after process if you are using `INITIALIZE_CLOB_OUTPUT` to write to a temporary `CLOB`.

Syntax

```
free_output;
```

Example

This example configures `APEX_JSON` for `CLOB` output, generate `JSON`, print the `CLOB` with `DBMS_OUTPUT`, and finally free the `CLOB`.

```
BEGIN
  apex_json.initialize_clob_output;

  apex_json.open_object;
  apex_json.write('hello', 'world');
  apex_json.close_object;

  dbms_output.put_line(apex_json.get_clob_output);

 apex_json.free_output;
 END;
```

15.9 FLUSH Procedure

This procedure flushes pending changes. Note that close procedures automatically flush.

Syntax

```
APEX_JSON.FLUSH
```

Parameters

None.

Example

This example writes incomplete JSON.

```
BEGIN
  apex_json.open_object;
  apex_json.write('attr', 'value');
  apex_json.flush;
  sys.htp.p('the "}" is missing');
END;
```

15.10 GET_BOOLEAN Function

This function returns a boolean number value.

Syntax

```
APEX_JSON.GET_BOOLEAN (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_default          IN BOOLEAN  DEFAULT NULL,
    p_values           IN t_values DEFAULT g_values )
RETURN BOOLEAN;
```

Parameters

Table 15-5 GET_BOOLEAN Function Parameters

Parameter	Description
`p_path`	Index into `p_values`.
`p[0-4]`	Each %N in `p_path` is replaced by `pN` and every i-th %s or %d is replaced by the `p[i-1]`.
`p_default`	The default value if the member does not exist.
`p_values`	Parsed JSON members. The default is `g_values`.

Returns

Table 15-6 GET_BOOLEAN Function Returns

Return	Description
`TRUE`	Value at the given path position.
`FALSE`	Value at the given path position.
`NULL`	Value at the given path position.

Table 15-6 (Cont.) GET_BOOLEAN Function Returns

Return	Description
VALUE_ERROR	Raises this error if p_values(p_path) is not boolean.

Example

This example parses a JSON string and prints the boolean value at a position.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": true } ] }');
    if apex_json.get_boolean(p_path=>'items[%d].foo', p0=>3,p_values=>j) then
        dbms_output.put_line('items[3].foo is true');
END IF;
END;
```

15.11 GET_CLOB_OUTPUT Function

Returns the temporary CLOB that you created with INITIALIZE_CLOB_OUTPUT.

Syntax

```
function get_clob_output
    return clob;
```

Example

This example configures APEX_JSON for CLOB output, generate JSON, print the CLOB with DBMS_OUTPUT, and finally free the CLOB.

```
BEGIN
  apex_json.initialize_clob_output;

  apex_json.open_object;
  apex_json.write('hello', 'world');
  apex_json.close_object;

  dbms_output.put_line(apex_json.get_clob_output);

 apex_json.free_output;
 END;
```

15.12 GET_COUNT Function

This function returns the number of array elements or object members.

Syntax

```
APEX_JSON.GET_COUNT (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
```

```
    p_values           IN t_values DEFAULT g_values )
RETURN NUMBER;
```

Parameters

Table 15-7 GET_COUNT Function Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Table 15-8 GET_COUNT Function Returns and Raised Errors

Return	Description
NUMBER	The number of array elements or object members or null if the array or object could not be found
VALUE_ERROR	Raises this error if p_values(p_path) is not an array or object.

Example

This example parses a JSON string and prints the number of members at positions.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    dbms_output.put_line(apex_json.get_count(p_path=>'.',p_values=>j)); -- 2 (foo
and bar)
    dbms_output.put_line(apex_json.get_count(p_path=>'bar',p_values=>j)); -- 4
END;
```

15.13 GET_DATE Function

This function returns a date member value.

Syntax

```
APEX_JSON.GET_DATE (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_default          IN DATE     DEFAULT NULL,
    p_format           IN VARCHAR2 DEFAULT c_date_iso8601,
    p_values           IN t_values DEFAULT g_values )
RETURN DATE;
```

Parameters

Table 15-9 GET_DATE Function Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_default	The default value if the member does not exist.
p_format	The date format mask.
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Table 15-10 GET_DATE Function Returns and Raised Errors

Return	Description
DATE	.Returns the date.
VALUE_ERROR	Raises this error if p_values(p_path) is not a date.

Example

This example parses a JSON string and prints the value at a position.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": "2014-04-29T10:08:00Z" }] }');

dbms_output.put_line(to_char(apex_json.get_date(p_path=>'items[%d].foo',p0=>3,
p_values=>j), 'DD-Mon-YYYY'));
END;
```

15.14 GET_MEMBERS Function

This function returns the table of OBJECT_MEMBERS names for an object.

Syntax

```
APEX_JSON.GET_MEMBERS (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_values           IN t_values DEFAULT g_values )
RETURN APEX_T_VARCHAR2;
```

Parameters

Table 15-11 GET_MEMBERS Function Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Table 15-12 GET_MEMBERS Function Returns and Raised Errors

Return	Description
OBJECT_MEMBERS	The OBJECT_MEMBERS of the object or null if the object could not be found.
VALUE_ERROR	Raises this error if p_values(p_path) is not an array or object.

Example

This example parses a JSON string and prints members at positions.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    dbms_output.put_line(apex_json.get_members(p_path=>'.',p_values=>j)(1)); -- foo
    dbms_output.put_line(apex_json.get_members(p_path=>'.',p_values=>j)(2)); -- bar
END;
```

15.15 GET_NUMBER Function

This function returns a numeric number value.

Syntax

```
APEX_JSON.GET_NUMBER (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_default          IN BOOLEAN DEFAULT NULL,
    p_values           IN t_values DEFAULT g_values )
RETURN NUMBER;
```

Parameters

Table 15-13 GET_NUMBER Function Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_default	The default value if the member does not exist.
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Table 15-14 GET_NUMBER Function Returns and Raised Errors

Return	Description
NUMBER	The value at the given path position.
VALUE_ERROR	Raises this error if p_values(p_path) is not a number.

Example

This example parses a JSON string and prints the value at a position.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_number(p_path=>'items[%d].foo',p0=>
3,p_values=>j));
END;
```

15.16 GET_VALUE Function

This function returns the t_value.

Syntax

```
APEX_JSON.GET_VALUE (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_values           IN t_values DEFAULT g_values )
RETURN t_value;
```

Parameters

Table 15-15 GET_VALUE Function Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Table 15-16 GET_VALUE Function Returns and Raised Errors

Return	Description
t_value	The t_value at the given path position. The record attributes are null if no data is found.
VALUE_ERROR	Raises this error if p_values(p_path) is not an array or object.

Example

This example parses a JSON string and prints attributes of values at positions.

```
DECLARE
    j apex_json.t_values;
    v apex_json.t_value;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    v := apex_json.get_value(p_path=>'bar[%d]',p0=> 2,p_values=>j); -- returns the
t_value for bar[2]
    dbms_output.put_line(v.number_value); -- 2
    v := apex_json.get_value(p_path=>'does.not.exist',p_values=>j);
    dbms_output.put_line(case when v.kind is null then 'not found!' end);
END;
```

15.17 GET_VARCHAR2 Function

This function returns a varchar2 member value. This function converts boolean and number values to varchar2 values.

Syntax

```
APEX_JSON.GET_VARCHAR2 (
    p_path             IN VARCHAR2,
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_default          IN BOOLEAN  DEFAULT NULL,
    p_values           IN t_values DEFAULT g_values )
RETURN VARCHAR2;
```

Parameters

Table 15-17 GET_VARCHAR2 Function Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_default	The default value if the member does not exist.
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Table 15-18 GET_VARCHAR2 Function Returns and Raised Errors

Return	Description
VARCHAR2	This is the value at the given path position.
VALUE_ERROR	Raises this error if p_values(p_path) is not an array or object.

Example

This example parses a JSON string and prints the value at a position.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_varchar2(p_path=>'items[%d].foo',p0=>
3,p_values=>j));
END;
```

15.18 GET_CLOB Function

This function returns clob member value. This function auto-converts varchar2, boolean and number values.

Syntax

```
get_clob (
p_path    in varchar2,
p0        in varchar2 default null,
p1        in varchar2 default null,
p2        in varchar2 default null,
p3        in varchar2 default null,
p4        in varchar2 default null,
p_default in clob default null,
p_values  in t_values default g_values )
return clob
```

Parameters

Table 15-19 GET_CLOB Function Parameters

Parameter	Description
p_values	Parsed json members. defaults to g_values.
p_path	Index into p_values.
p[0-4]	Each %N in p_path will be replaced by pN and every i-th %s or %d will be replaced by the p[i-1].
p_default	Default value if the member does not exist.

Returns/Raised Errors

Table 15-20 GET_CLOB Function Returns and Raised Errors

Return/Raised Errors	Description
a clob	Value at the given path position
VALUE_ERROR	If p_values(p_path) is an array or an object

Example

Parse a JSON string and print the value at a position.

```
declare
    j apex_json.t_values;
begin
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_clob (
    p_values => j,
    p_path => 'items[%d].foo',
    p0 => 3));
end;
```

15.19 INITIALIZE_CLOB_OUTPUT Procedure

This procedure initializes the output interface to write to a temporary CLOB. The default is to write to SYS.HTP. If using CLOB output, you should call FREE_OUTPUT() at the end to free the CLOB.

Syntax

```
APEX_JSON.INITIALIZE_CLOB_OUTPUT (
    p_dur         in pls_integer default sys.dbms_lob.call,
    p_cache       in boolean     default true,
    p_indent      in pls_integer default null );
```

Parameters

Table 15-21 INITIALIZE_CLOB_OUTPUT Procedure Parameters

Parameter	Description
p_dur	Duration of the temporary CLOB. this can be DBMS_LOB.SESSION or DBMS_LOB.CALL (the default).
p_cache	Specifies if the lob should be read into buffer cache or not.
p_indent	Indent level. Defaults to 2 if debug is turned on, 0 otherwise.

Example

This example configures APEX_JSON for CLOB output, generate JSON, print the CLOB with DBMS_OUTPUT, and finally free the CLOB.

```
BEGIN
  apex_json.initialize_clob_output;

  apex_json.open_object;
  apex_json.write('hello', 'world');
  apex_json.close_object;

  dbms_output.put_line(apex_json.get_clob_output);

 apex_json.free_output;
 END;
```

15.20 INITIALIZE_OUTPUT Procedure

This procedure initializes the output interface. You only have to call this procedure if you want to modify the parameters below. Initially, output is already configured with the defaults mentioned in the parameter table.

Syntax

```
APEX_JSON.INITIALIZE_OUTPUT (
    p_http_header     in boolean     default true,
    p_http_cache      in boolean     default false,
    p_http_cache_etag in varchar2    default null,
    p_indent          in pls_integer default null );
```

Parameters

Table 15-22 INITIALIZE_OUTPUT Procedure Parameters

Parameter	Description
p_http_header	If TRUE (the default), write an application/JSON mime type header.
p_http_cache	This parameter is only relevant if p_write_header is TRUE. If TRUE, writes Cache-Control: max-age=315360000. If FALSE (the default), writes Cache-Control: no-cache. Otherwise, does not write Cache-Control.

Table 15-22 (Cont.) INITIALIZE_OUTPUT Procedure Parameters

Parameter	Description
http_cache_etag	If not null, writes an etag header. This parameter is only used if P_HTTP_CACHE is true.
p_indent	Indent level. Defaults to 2, if debug is turned on, otherwise defaults to 0.

Example

This example configures APEX_JSON to not emit default headers, because they are written directly.

```
BEGIN
  apex_json.initialize_output (
      p_http_header => false );

  sys.owa_util.mime_header('application/json', false);
  sys.owa_util.status_line(429, 'Too Many Requests');
  sys.owa_util.http_header_close;
  --
  apex_json.open_object;
  apex_json.write('maxRequestsPerSecond', 10);
  apex_json.close_object;
END;
```

15.21 OPEN_ARRAY Procedure

This procedure writes an open bracket symbol as follows:

```
[
```

Syntax

```
APEX_JSON.OPEN_ARRAY (
    p_name     IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 15-23 OPEN_ARRAY Procedure Parameters

Parameter	Description
p_name	If not null, write an object attribute name and colon before the opening bracket.

Example

This example performs a `write { "array":[ 1 ,[ ] ] }`.

```
BEGIN
  apex_json.open_object; -- {
  apex_json.open_array('array'); -- "array": [
  apex_json.write(1); -- 1
  apex_json.open_array; -- , [
  apex_json.close_array; -- ]
```

```
  apex_json.close_array; -- ]
  apex_json.close_object; -- }
END;
```

15.22 OPEN_OBJECT Procedure

This procedure writes an open curly bracket symbol as follows:

```
{
```

Syntax

```
APEX_JSON.OPEN_OBJECT (
    p_name     IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 15-24 OPEN_OBJECT Procedure Parameters

Parameter	Description
p_name	If not null, write an object attribute name and colon before the opening brace.

Example

This example performs a `write { "obj": { "obj-attr": "value" }}`.

```
BEGIN
  apex_json.open_object; -- {
  apex_json.open_object('obj'); -- "obj": {
  apex_json.write('obj-attr', 'value'); -- "obj-attr": "value"
  apex_json.close_all; -- }}
END;
```

15.23 PARSE Procedure Signature 1

This procedure parses a JSON-formatted `varchar2` or `clob` and puts the members into `p_values`.

Syntax

```
APEX_JSON.PARSE (
    p_values   in out nocopy t_values,
    p_source   in varchar2,
    p_strict   in boolean default true );

APEX_JSON.PARSE (
    p_values   in out nocopy t_values,
    p_source   in clob,
    p_strict   in boolean default true );
```

Parameters

Table 15-25 PARSE Procedure Parameters

Parameter	Description
p_values	An index by varchar2 result array which contains the JSON members and values. The default is g_values.
p_source	The JSON source (varchar2 or clob)
p_strict	If TRUE (default), enforce strict JSON rules

Example

This example parses JSON and prints member values.

```
DECLARE
    l_values apex_json.t_values;
BEGIN
    apex_json.parse (
        p_values => l_values,
        p_source => '{ "type": "circle", "coord": [10, 20] }' );
    sys.htp.p('Point at '||
        apex_json.get_number (
            p_values => l_values,
            p_path   => 'coord[1]')||
        ','||
        apex_json.get_number (
            p_values => l_values,
            p_path   => 'coord[2]'));
END;
```

15.24 PARSE Procedure Signature 2

This procedure parses a JSON-formatted varchar2 or clob and puts the members into the package global g_values. This simplified API works similar to the parse() procedure for signature 1, but saves the developer from declaring a local variable for parsed JSON data and passing it to each JSON API call.

Syntax

```
APEX_JSON.PARSE (
    p_source   IN VARCHAR2,
    p_strict   IN BOOLEAN DEFAULT TRUE );

APEX_JSON.PARSE (
    p_source   IN CLOB,
    p_strict   IN BOOLEAN DEFAULT TRUE );
```

Parameters

Table 15-26 PARSE Procedure Parameters

Parameter	Description
p_source	The JSON source (varchar2 or clob).
p_strict	If TRUE (default), enforce strict JSON rules.

Example

This example parses JSON and prints member values.

```
apex_json.parse('{ "type": "circle", "coord": [10, 20] }');
sys.htp.p('Point at '||
    apex_json.get_number(p_path=>'coord[1]')||
    ','||
    apex_json.get_number(p_path=>'coord[2]'));
```

15.25 STRINGIFY Function Signature 1

This function converts a string to an escaped JSON value.

Syntax

```
APEX_JSON.STRINGIFY (
    p_value  IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Table 15-27 STRINGIFY Function Parameters

Parameter	Description
p_value	The string to be converted.

Returns

Table 15-28 STRINGIFY Function Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example is a query that returns a JSON `varchar2` value.

```
select apex_json.stringify('line 1'||chr(10)||'line 2') from dual;
```

15.26 STRINGIFY Function Signature 2

This function converts a number to an escaped JSON value.

Syntax

```
APEX_JSON.STRINGIFY (
    p_value  IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Table 15-29 STRINGIFY Function Parameters

Parameter	Description
p_value	The number to be converted.

Returns

Table 15-30 STRINGIFY Function Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example is a query that returns a JSON number value.

```
select apex_json.stringify(-1/10) from dual
```

15.27 STRINGIFY Function Signature 3

This function converts a date to an escaped JSON value.

Syntax

```
APEX_JSON.STRINGIFY (
    p_value  IN DATE,
    p_format IN VARCHAR2 DEFAULT c_date_iso8601 )
RETURN VARCHAR2;
```

Parameters

Table 15-31 STRINGIFY Function Parameters

Parameter	Description
p_value	The date value to be converted.

Returns

Table 15-32 STRINGIFY Function Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example is a query that returns a JSON `varchar2` value that is suitable to be converted to dates.

```
select apex_json.stringify(sysdate) from dual
```

15.28 STRINGIFY Function Signature 4

This function converts a boolean value to an escaped JSON value.

Syntax

```
APEX_JSON.STRINGIFY (
    p_value  IN BOOLEAN,
RETURN VARCHAR2;
```

Parameters

Table 15-33 STRINGIFY Function Parameters

Parameter	Description
p_value	The boolean value to be converted.

Returns

Table 15-34 STRINGIFY Function Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example demonstrates printing JSON boolean values.

```
BEGIN
  sys.htp.p(apex_json.stringify(true));
  sys.htp.p(apex_json.stringify(false));
END;
```

15.29 TO_XMLTYPE Function

This procedure parses a JSON-formatted `varchar2` or `CLOB` and converts it to an xmltype.

Syntax

```
APEX_JSON.TO_XMLTYPE (
    p_source   IN VARCHAR2,
    p_strict   IN BOOLEAN DEFAULT TRUE )
RETURN sys.xmltype;

APEX_JSON.TO_XMLTYPE (
    p_source   IN CLOB,
    p_strict   IN BOOLEAN DEFAULT TRUE )
RETURN sys.xmltype;
```

Parameters

Table 15-35 TO_XMLTYPE Function Parameters

Parameter	Description
p_source	The JSON source (VARCHAR2 or CLOB)
p_strict	If TRUE (default), enforce strict JSON rules

Returns

Table 15-36 TO_XMLTYPE Function Returns

Return	Description
sys.xmltype	An xmltype representation of the JSON data.

Example

This example parses JSON and prints the XML representation.

```
DECLARE
    l_xml xmltype;
BEGIN
    l_xml := apex_json.to_xmltype('{ "items": [ 1, 2, { "foo": true } ] }');
    dbms_output.put_line(l_xml.getstringval);
END;
```

15.30 WRITE Procedure Signature 1

This procedure writes an array attribute of type VARCHAR2.

Syntax

```
APEX_JSON.WRITE (
    p_value    IN VARCHAR2 );
```

Parameters

Table 15-37 WRITE Procedure Parameters

Parameter	Description
p_value	The value to be written.

Example

This example writes an array containing 1, "two", "long text", false, the current date and a JSON representation of an xml document.

```
DECLARE
  l_clob clob := 'long text';
  l_xml sys.xmltype := sys.xmltype('<obj><foo>1</foo><bar>2</bar></obj>');
BEGIN
  apex_json.open_array; -- [
  apex_json.write(1); -- 1
  apex_json.write('two'); -- , "two"
```

```
  apex_json.write(l_clob); -- , "long text"
  apex_json.write(false); -- , false
  apex_json.write(sysdate); -- , "2014-05-05T05:36:08Z"
  apex_json.write(localtimestamp); -- , "2014-05-05T05:36:08.5434Z"
  apex_json.write(current_timestamp); -- , "2014-05-05T05:36:08.5434+02:00"
  apex_json.write(l_xml); -- , { "foo": 1, "bar": 2 }
  apex_json.close_array; -- ]
END;
```

15.31 WRITE Procedure Signature 2

This procedure writes an array attribute. of type `clob`.

Syntax

```
APEX_JSON.WRITE (
    p_value    IN CLOB );
```

Parameters

Table 15-38 WRITE Procedure Parameters

Parameter	Description
p_value	The value to be written.

Example

See "WRITE Procedure Signature 1 (page 15-24)".

15.32 WRITE Procedure Signature 3

This procedure writes an array attribute of type `NUMBER`.

Syntax

```
APEX_JSON.WRITE (
    p_value    IN NUMBER );
```

Parameters

Table 15-39 WRITE Procedure Parameters

Parameter	Description
p_value	The value to be written.

Example

See "WRITE Procedure Signature 1 (page 15-24)".

15.33 WRITE Procedure Signature 4

This procedure writes an array attribute. of type date

Syntax

```
APEX_JSON.WRITE (
    p_value    IN DATE,
    p_format   IN VARCHAR2 DEFAULT c_date_iso8601 );
```

Parameters

Table 15-40 WRITE Procedure Parameters

Parameter	Description
p_value	The value to be written.
p_format	The date format mask (default c_date_iso8601).

Example

See "WRITE Procedure Signature 1 (page 15-24)".

15.34 WRITE Procedure Signature 5

This procedure writes an array attribute of type boolean.

Syntax

```
APEX_JSON.WRITE (
    p_value    IN BOOLEAN );
```

Parameters

Table 15-41 WRITE Procedure Parameters

Parameter	Description
p_value	The value to be written.

Example

See "WRITE Procedure Signature 1 (page 15-24)".

15.35 WRITE Procedure Signature 6

This procedure writes an array attribute of type sys.xmltype. The procedure uses a XSL transformation to generate JSON. To determine the JSON type of values, it uses the following rules:

- If the value is empty, it generates a NULL value.
- If upper(value) is TRUE, it generates a boolean true value.
- If upper(value) is FALSE, it generates a boolean false value.
- If the XPath number function returns TRUE, it emits the value as is. Otherwise, it enquotes the value (that is, treats it as a JSON string).

Syntax

```
APEX_JSON.WRITE (
    p_value    IN sys.xmltype );
```

Parameters

Table 15-42 WRITE Procedure Parameters

Parameter	Description
p_value	The value to be written.

Example

See "WRITE Procedure Signature 1 (page 15-24)".

15.36 WRITE Procedure Signature 7

This procedure writes an array with all rows that the cursor returns. Each row is a separate object. If the query contains object type, collection, or cursor columns, the procedure uses `write(xmltype)` to generate JSON. Otherwise, it uses `DBMS_SQL` to fetch rows and the `write()` procedures for the appropriate column data types for output. If the column type is `varchar2` and the uppercase value is `'TRUE'` or `'FALSE'`, it generates boolean values.

Syntax

```
APEX_JSON.WRITE (
    p_cursor      IN OUT NOCOPY sys_refcursor );
```

Parameters

Table 15-43 WRITE Procedure Parameters

Parameter	Description
p_cursor	The cursor.

Example 1

This example writes an array containing JSON objects for departments 10 and 20.

```
DECLARE
    c sys_refcursor;
BEGIN
    open c for select deptno, dname, loc from dept where deptno in (10, 20);
    apex_json.write(c);
END;
```

This is the output:

```
[ { "DEPTNO":10 ,"DNAME":"ACCOUNTING" ,"LOC":"NEW YORK" }
, { "DEPTNO":20 ,"DNAME":"RESEARCH" ,"LOC":"DALLAS" } ]
```

15.37 WRITE Procedure Signature 8

This procedure writes an object attribute of type VARCHAR2.

Syntax

```
APEX_JSON.WRITE (
    p_name        IN VARCHAR2,
    p_value       IN VARCHAR2,
    p_write_null  IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-44 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If true, write NULL values. If false (the default), do not write NULLs.

Example

This example writes an object with named member attributes of various types. The comments to the right of the statements show the output that they generate.

```
DECLARE
  l_clob clob := 'long text';
  l_xml sys.xmltype := sys.xmltype('<obj><foo>1</foo><bar>2</bar></obj>');
BEGIN
  apex_json.open_object; -- {
  apex_json.write('a1', 1); -- "a1": 1
  apex_json.write('a2', 'two'); -- ,"a2": "two"
  apex_json.write('a3', l_clob); -- ,"a3": "long text"
  apex_json.write('a4', false); -- ,"a4": false
  apex_json.write('a5', sysdate); -- ,"a5": "2014-05-05T05:36:08Z"
  apex_json.write('a6', l_xml); -- ,"a6": { "foo": 1, "bar": 2 }
  apex_json.close_object; -- }
END;
```

15.38 WRITE Procedure Signature 9

This procedure writes an object attribute of type CLOB.

Syntax

```
APEX_JSON.WRITE (
    p_name         IN VARCHAR2,
    p_value        IN CLOB,
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-45 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If true, write NULL values. If false (the default), do not write NULLs.

Example

See example for "WRITE Procedure Signature 8 (page 15-28)".

15.39 WRITE Procedure Signature 10

This procedure writes an object attribute of type NUMBER.

Syntax

```
APEX_JSON.WRITE (
    p_name         IN VARCHAR2,
    p_value        IN NUMBER,
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-46 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If true, write NULL values. If false (the default), do not write NULLs.

Example

See example for "WRITE Procedure Signature 8 (page 15-28)".

15.40 WRITE Procedure Signature 11

This procedure writes an object attribute of type date.

Syntax

```
APEX_JSON.WRITE (
    p_name         IN VARCHAR2,
    p_value        IN DATE,
    p_format       IN VARCHAR2 DEFAULT c_date_iso8691,
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-47 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_format	The date format mask (default apex_json.c_date_iso8601.
p_write_null	If true, write NULL values. If false (the default), do not write NULL.

Example

See example for "WRITE Procedure Signature 8 (page 15-28)".

15.41 WRITE Procedure Signature 12

This procedure writes an object attribute of type boolean.

Syntax

```
APEX_JSON.WRITE (
    p_name         IN VARCHAR2,
    p_value        IN BOOLEAN,
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-48 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If true, write NULL values. If false (the default), do not write NULL.

Example

See example for "WRITE Procedure Signature 8 (page 15-28)".

15.42 WRITE Procedure Signature 13

This procedure writes an attribute where the value is an array that contains all rows that the cursor returns. Each row is a separate object.

If the query contains object type, collection, or cursor columns, the procedure uses write(p_name,<xmltype>). See "WRITE Procedure Signature 14 (page 15-31)". Otherwise, it uses DBMS_SQL to fetch rows and the write() procedures for the appropriate column data types for output. If the column type is varchar2 and the uppercase value is 'TRUE' or 'FALSE', it generates boolean values.

Syntax

```
APEX_JSON.WRITE (
    p_name        IN VARCHAR2,
    p_cursor      IN OUT NOCOPY sys_refcursor );
```

Parameters

Table 15-49 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_cursor	The cursor.

Example

This example writes an array containing JSON objects for departments 10 and 20, as an object member attribute.

```
DECLARE
  c sys_refcursor;
BEGIN
  open c for select deptno,
                    dname,
                    cursor(select empno,
                                  ename
                             from emp e
                            where e.deptno=d.deptno) emps
               from dept d;
  apex_json.open_object;
  apex_json. write('departments', c);
  apex_json.close_object;
END;

{ "departments":[
      {"DEPTNO":10,
       "DNAME":"ACCOUNTING",
       "EMPS":[{"EMPNO":7839,"ENAME":"KING"}]},
      ...
     ,{"DEPTNO":40,"DNAME":"OPERATIONS","EMPS":null}] }
```

15.43 WRITE Procedure Signature 14

This procedure writes an array attribute of type `sys.xmltype`. The procedure uses a XSL transformation to generate JSON. To determine the JSON type of values, it uses the following rules:

- If the value is empty, it generates a `NULL` value.
- If upper(value) is `TRUE`, it generates a boolean true value.
- If upper(value) is `FALSE`, it generates a boolean false value.
- If the `XPath` number function returns true, it emits the value as is. Otherwise, it enquotes the value (that is, treats it as a JSON string).

Syntax

```
APEX_JSON.WRITE (
    p_name         IN VARCHAR2,
    p_value        IN sys.xmltype,
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-50 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_value	The value to be written. The XML is converted to JSON
p_write_null	If true, write NULL values. If false (the default), do not write NULLs.

Example

See example for "WRITE Procedure Signature 13 (page 15-30)".

15.44 WRITE Procedure Signature 15

This procedure writes parts of a parsed APEX_JSON.t_values table.

Syntax

```
APEX_JSON.WRITE (
    p_values           IN t_values,
    p_path             IN VARCHAR2 DEFAULT '.',
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 15-51 WRITE Procedure Parameters

Parameter	Description
p_values	The parsed JSON members.
p_path	The index into p_values.
p[0-4]	Each %N in p_path will be replaced by pN and every i-th %s or %d is replaced by p[i-1].

Example

This example parses a JSON string and writes parts of it.

```
DECLARE
  j apex_json.t_values;
BEGIN
```

```
  apex_json.parse(j, '{ "foo": 3, "bar": { "x": 1, "y": 2 }}');
  apex_json.write(j,'bar');
END;
```

15.45 WRITE Procedure Signature 16

This procedure writes parts of a parsed APEX_JSON.t_values table as an object member attribute.

Syntax

```
APEX_JSON.WRITE (
    p_name             IN VARCHAR2,
    p_values           IN t_values,
    p_path             IN VARCHAR2 DEFAULT '.',
    p0                 IN VARCHAR2 DEFAULT NULL,
    p1                 IN VARCHAR2 DEFAULT NULL,
    p2                 IN VARCHAR2 DEFAULT NULL,
    p3                 IN VARCHAR2 DEFAULT NULL,
    p4                 IN VARCHAR2 DEFAULT NULL,
    p_write_null       IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 15-52 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_values	The parsed JSON members.
p_path	The index into p_values.
p[0-4]	Each %N in p_path will be replaced by pN and every i-th %s or %d is replaced by p[i-1].
p_write_null	If true, write NULL values. If false (the default), do not write NULLs.

Example

This example parses a JSON string and writes parts of it as an object member.

```
DECLARE
  j apex_json.t_values;
BEGIN
  apex_json.parse(j, '{ "foo": 3, "bar": { "x": 1, "y": 2 }}');
  apex_json.open_object; -- {
  apex_json.write('parsed-bar',j,'bar');-- "parsed-bar":{ "x":1 ,"y":2 }
  apex_json.close_object; -- }
END;
```

15.46 WRITE Procedure Signature 17

This procedure writes an array attribute of type VARCHAR2.

Syntax

```
procedure write (
      p_name        in varchar2,
      p_values      in apex_t_varchar2,
      p_write_null  in boolean default false );
```

Parameters

Table 15-53 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_values	The VARCHAR2 array values to be written.
p_write_null	If true, write an empty array. If false (the default), do not -- write an empty array.

Example

This example writes an array containing a, b, c.

```
declare
 l_values apex_t_varchar2 := apex_t_varchar2( 'a', 'b', 'c' );
begin
  apex_json.open_object;                -- {
  apex_json.write('array', l_values );  --   "array": [ "a", "b", "c" ]
  apex_json.close_object;               -- }
end;
```

15.47 WRITE Procedure Signature 18

This procedure writes an array attribute of type NUMBER .

Syntax

```
procedure write (
   p_name        in varchar2,
   p_values      in apex_t_number,
   p_write_null  in boolean default false );
```

Parameters

Table 15-54 WRITE Procedure Parameters

Parameter	Description
p_name	The attribute name.
p_values	The NUMBER array values to be written.
p_write_null	If true, write an empty array. If false (the default), do not -- write an empty array.

Example

This example writes an array containing 1, 2, 3.

```
declare
  l_values apex_t_number := apex_t_number( 1, 2, 3 );
begin
  apex_json.open_object;                -- {
  apex_json.write('array', l_values );  --   "array": [ 1, 2, 3 ]
  apex_json.close_object;               -- }
end;
```

16

APEX_LANG

You can use APEX_LANG API to translate messages.

16.1 CREATE_LANGUAGE_MAPPING Procedure

Use this procedure to create the language mapping for the translation of an application. Translated applications are published as new applications, but are not directly editable in the App Builder.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

```
APEX_LANG.CREATE_LANGUAGE_MAPPING (
  p_application_id IN NUMBER,
  p_language IN VARCHAR2,
  p_translation_application_id IN NUMBER )
```

Parameters

Table 16-1 CREATE_LANGUAGE_MAPPING Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to create the language mapping. This is the ID of the primary language application.

Table 16-1 (Cont.) CREATE_LANGUAGE_MAPPING Parameters

Parameter	Description
p_language	The IANA language code for the mapping. Examples include en-us, fr-ca, ja, he.
p_translation_application_id	Unique integer value for the ID of the underlying translated application. This number cannot end in 0.

Example

The following example demonstrates the creation of the language mapping for an existing Application Express application.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;

    -- Now, actually create the language mapping
    apex_lang.create_language_mapping(
        p_application_id => 63969,
        p_language => 'ja',
        p_translation_application_id => 778899 );
    commit;
    --
    -- Print what we just created to confirm
    --
    for c1 in (select *
                 from apex_application_trans_map
                where primary_application_id = 63969) loop
        dbms_output.put_line( 'translated_application_id: ' ||
c1.translated_application_id );
        dbms_output.put_line( 'translated_app_language: ' ||
c1.translated_app_language );
    end loop;
end;
/
```

16.2 DELETE_LANGUAGE_MAPPING Procedure

Use this procedure to delete the language mapping for the translation of an application. This procedure deletes all translated strings in the translation repository for the specified language and mapping. Translated applications are published as new applications, but are not directly editable in the App Builder.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

```
APEX_LANG.DELETE_LANGUAGE_MAPPING (
  p_application_id IN NUMBER,
  p_language IN VARCHAR2 )
```

Parameters

Table 16-2 DELETE_LANGUAGE_MAPPING Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to delete the language mapping. This is the ID of the primary language application.
p_language	The IANA language code for the existing mapping. Examples include en-us, fr-ca, ja, he.

Example

The following example demonstrates the deletion of the language mapping for an existing Application Express application and existing translation mapping.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                 from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
     end loop;
    -- Now, delete the language mapping
    apex_lang.delete_language_mapping(
        p_application_id => 63969,
        p_language => 'ja' );
    commit;
    --
    -- Print what we just updated to confirm
    --
    for c1 in (select count(*) thecount
                 from apex_application_trans_map
                where primary_application_id = 63969) loop
        dbms_output.put_line( 'Translation mappings found: ' || c1.thecount );
    end loop;
end;
/
```

16.3 LANG Function

Use this function to return a translated text string for translations defined in dynamic translations.

Syntax

```
APEX_LANG.LANG (
    p_primary_text_string IN VARCHAR2 DEFAULT NULL,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    ...
    p9 IN VARCHAR2 DEFAULT NULL,
    p_primary_language IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 16-3 LANG Parameters

Parameter	Description
`p_primary_text_string`	Text string of the primary language. This is the value of the Translate From Text in the dynamic translation.
`p0` through `p9`	Dynamic substitution value: p0 corresponds to %0 in the translation string; p1 corresponds to %1 in the translation string; p2 corresponds to %2 in the translation string, and so on.
`p_primary_language`	Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. See also: Specifying the Primary Language for an Application in the *Oracle Application Express App Builder User's Guide*.

Example

Suppose you have a table that defines all primary colors. You could define a dynamic message for each color and then apply the LANG function to the defined values in a query. For example:

```
SELECT APEX_LANG.LANG(color)
FROM my_colors
```

If you were running the application in German, RED was a value for the color column in the `my_colors` table, and you defined the German word for red, the previous example would return ROT.

16.4 MESSAGE Function

Use this function to translate text strings (or messages) generated from PL/SQL stored procedures, functions, triggers, packaged procedures, and functions.

Syntax

```
APEX_LANG.MESSAGE (
    p_name            IN VARCHAR2 DEFAULT NULL,
    p0                IN VARCHAR2 DEFAULT NULL,
    p1                IN VARCHAR2 DEFAULT NULL,
    p2                IN VARCHAR2 DEFAULT NULL,
    ...
    p9                IN VARCHAR2 DEFAULT NULL,
    p_lang            IN VARCHAR2 DEFAULT NULL,
    p_application_id  IN NUMBER   DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 16-4 MESSAGE Parameters

Parameter	Description
p_name	Name of the message as defined in Text Messages under Shared Components of your application in Oracle Application Express.
p0 through p9	Dynamic substitution value: p0 corresponds to %0 in the translation string; p1 corresponds to %1 in the translation string; p2 corresponds to %2 in the translation string, and so on.
p_lang	Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. See also: Specifying the Primary Language for an Application in the *Oracle Application Express App Builder User's Guide*.
p_application_id	Used to specify the application ID within the current workspace that owns the translated message you wish to return. Useful when coding packages that might be called outside of the scope of Oracle Application Express such as packages called from a database job.

Example

The following example assumes you have defined a message called GREETING_MSG in your application in English as "Good morning %0" and in German as "Guten Tag %1". The following example demonstrates how you could invoke this message from PL/SQL:

```
BEGIN
--
-- Print the greeting
--
HTP.P(APEX_LANG.MESSAGE('GREETING_MSG', V('APP_USER')));
END;
```

How the p_lang attribute is defined depends on how the Application Express engine derives the Application Primary Language. For example, if you are running the application in German and the previous call is made to the APEX_LANG.MESSAGE API, the Application Express engine first looks for a message called GREETING_MSG with a LANG_CODE of de. If it does not find anything, then it is reverted to the Application Primary Language attribute. If it still does not find anything, the

Application Express engine looks for a message by this name with a language code of en.

See also:

Specifying the Primary Language for an Application in the *Oracle Application Express App Builder User's Guide*.

16.5 PUBLISH_APPLICATION Procedure

Use this procedure to publish the translated version of an application. This procedure creates an underlying, hidden replica of the primary application and merges the strings from the translation repository in this new application. Perform a seed and publish process each time you want to update the translated version of your application and synchronize it with the primary application.

This application is not visible in the App Builder. It can be published and exported, but not directly edited.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

```
APEX_LANG.PUBLISH_APPLICATION (
  p_application_id IN NUMBER,
  p_language IN VARCHAR2 )
```

Parameters

Table 16-5 PUBLISH_APPLICATION Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to publish and create the translated version. This is the ID of the primary language application.
p_language	The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he.

Example

The following example demonstrates the publish process for an Application Express application and language.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
```

```
    for c1 in (select workspace_id
                from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Now, publish the translated version of the application
    apex_lang.publish_application(
        p_application_id => 63969,
        p_language => 'ja' );
    commit;
end;
/
```

16.6 SEED_TRANSLATIONS Procedure

Use this procedure to seed the translation repository for the specified application and language. This procedure populates the translation repository with all of the new, updated and removed translatable strings from your application. Perform a seed and publish process each time you want to update the translated version of your application and synchronize it with the primary application.

Syntax

```
APEX_LANG.SEED_TRANSLATIONS (
  p_application_id IN NUMBER,
  p_language IN VARCHAR2 )
```

Parameters

Table 16-6 SEED_TRANSLATIONS Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to update the translation repository. This is the ID of the primary language application.
p_language	The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he.

Example

The following example demonstrates the seeding process of the translation repository for an Application Express application and language.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Now, seed the translation repository
```

```
    apex_lang.seed_translations(
        p_application_id => 63969,
        p_language => 'ja' );
    commit;
    -- Print out the total number of potentially translatable strings
    --
    for c1 in (select count(*) thecount
                 from apex_application_trans_repos
                where application_id = 63969) loop
        dbms_output.put_line( 'Potentially translatable strings found: ' ||
c1.thecount );
    end loop;
end;
/
```

16.7 UPDATE_LANGUAGE_MAPPING Procedure

Use this procedure to update the language mapping for the translation of an application. Translated applications are published as new applications, but are not directly editable in the App Builder.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

```
APEX_LANG.UPDATE_LANGUAGE_MAPPING (
  p_application_id IN NUMBER,
  p_language IN VARCHAR2,
  p_new_trans_application_id IN NUMBER )
```

Parameters

Table 16-7 UPDATE_LANGUAGE_MAPPING Parameters

Parameters	Description
p_application_id	The ID of the application for which you want to update the language mapping. This is the ID of the primary language application.
p_language	The IANA language code for the existing mapping. Examples include en-us, fr-ca, ja, he. The language of the mapping cannot be updated with this procedure, only the new translation application ID.
p_new_trans_application_id	New unique integer value for the ID of the underlying translated application. This number cannot end in 0.

Example

The following example demonstrates the update of the language mapping for an existing Application Express application and existing translation mapping.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
```

```
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Now, update the language mapping
    apex_lang.update_language_mapping(
        p_application_id => 63969,
        p_language => 'ja',
        p_new_trans_application_id => 881188 );
    commit;
    --
    -- Print what we just updated to confirm
    --
    for c1 in (select *
                from apex_application_trans_map
               where primary_application_id = 63969) loop
        dbms_output.put_line( 'translated_application_id: ' ||
c1.translated_application_id );
        dbms_output.put_line( 'translated_app_language: ' ||
c1.translated_app_language );
    end loop;
end;
/
```

16.8 UPDATE_MESSAGE Procedure

Use this procedure to update a translatable text message for the specified application.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

```
APEX_LANG.UPDATE_MESSAGE (
  p_id IN NUMBER,
  p_message_text IN VARCHAR2 )
```

Parameters

Table 16-8 UPDATE_MESSAGE Parameters

Parameter	Description
p_id	The ID of the text message.
p_message_text	The new text for the translatable text message.

Example

The following example demonstrates an update of an existing translatable text message.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                 from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Locate the ID of the specific message and update it with the new text
    for c1 in (select translation_entry_id
                 from apex_application_translations
                where application_id = 63969
                  and translatable_message = 'TOTAL_COST'
                  and language_code = 'ja') loop
        apex_lang.update_message(
            p_id => c1.translation_entry_id,
            p_message_text => 'The total cost is: %0');
        commit;
        exit;
    end loop;
end;
/
```

16.9 UPDATE_TRANSLATED_STRING Procedure

Use this procedure to update a translated string in the seeded translation repository.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

```
APEX_LANG.UPDATE_TRANSLATED_STRING (
  p_id IN NUMBER,
  p_language IN VARCHAR2
  p_string IN VARCHAR2 )
```

Parameters

Table 16-9 UPDATE_TRANSLATED_STRING Parameters

Parameter	Description
p_id	The ID of the string in the translation repository.

Table 16-9 (Cont.) UPDATE_TRANSLATED_STRING Parameters

Parameter	Description
p_language	The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he. The language of the mapping cannot be updated with this procedure, only the new translation application ID.
p_string	The new value for the string in the translation repository.

Example

The following example demonstrates an update of an existing string in the translation repository.

```
begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema. The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                 from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Locate all strings in the repository for the specified application
    -- which are 'Search' and change to 'Find'
    for c1 in (select id
                 from apex_application_trans_repos
                where application_id = 63969
                  and dbms_lob.compare(from_string, to_nclob('Search')) = 0
                  and language_code = 'ja') loop
        apex_lang.update_translated_string(
            p_id => c1.id,
            p_language => 'ja',
            p_string => 'Find');
        commit;
        exit;
    end loop;
end;
/
```

17

APEX_LDAP

You can use APEX_LDAP to perform various operations related to Lightweight Directory Access Protocol (LDAP) authentication.

17.1 AUTHENTICATE Function

The AUTHENTICATE function returns a boolean TRUE if the user name and password can be used to perform a SIMPLE_BIND_S, call using the provided search base, host, and port.

Syntax

```
APEX_LDAP.AUTHENTICATE(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_password     IN VARCHAR2 DEFAULT NULL,
    p_search_base  IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN BOOLEAN;
```

Parameters

Table 17-1 AUTHENTICATE Parameters

Parameter	Description
p_username	Login name of the user.
p_password	Password for p_username.
p_search_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.

Table 17-1 (Cont.) AUTHENTICATE Parameters

Parameter	Description
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).

Example

The following example demostrates how to use the APEX_LDAP.AUTHENTICATE function to verify user credentials against an LDAP Server.

```
IF APEX_LDAP.AUTHENTICATE(
    p_username =>'firstname.lastname',
    p_password =>'abcdef',
    p_search_base => 'cn=user,l=amer,dc=my_company,dc=com',
    p_host => 'our_ldap_sever.my_company.com',
    p_port => 389) THEN
    dbms_output.put_line('authenticated');
ELSE
    dbms_output.put_line('authentication failed');
END IF;
```

17.2 GET_ALL_USER_ATTRIBUTES Procedure

The GET_ALL_USER_ATTRIBUTES procedure returns two OUT arrays of user_attribute names and values for the user name designated by p_username (with password if required) using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2 DEFAULT NULL,
    p_host              IN VARCHAR2,
    p_port              IN VARCHAR2 DEFAULT 389,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
    p_attributes        OUT apex_application_global.vc_arr2,
    p_attribute_values  OUT apex_application_global.vc_arr2);
```

Parameters

Table 17-2 GET_ALL_USER_ATTRIBUTES Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.

Table 17-2 (Cont.) GET_ALL_USER_ATTRIBUTES Parameters

Parameter	Description
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).
p_attributes	An array of attribute names returned.
p_attribute_values	An array of values returned for each corresponding attribute name returned in p_attributes.

Example

The following example demonstrates how to use the APEX_LDAP.GET_ALL_USER_ATTRIBUTES procedure to retrieve all attribute value's associated to a user.

```
DECLARE
    L_ATTRIBUTES       apex_application_global.vc_arr2;
    L_ATTRIBUTE_VALUES apex_application_global.vc_arr2;
BEGIN
    APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
        p_username         => 'firstname.lastname',
        p_pass             => 'abcdef',
        p_auth_base        => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host             => 'our_ldap_sever.my_company.com',
        p_port             => '389',
        p_attributes       => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);

     FOR i IN L_ATTRIBUTES.FIRST..L_ATTRIBUTES.LAST LOOP
         htp.p('attribute name: '||L_ATTRIBUTES(i));
         htp.p('attribute value: '||L_ATTRIBUTE_VALUES(i));
     END LOOP;
END;
```

17.3 GET_USER_ATTRIBUTES Procedure

The GET_USER_ATTRIBUTES procedure returns an OUT array of user_attribute values for the user name designated by p_username (with password if required) corresponding to the attribute names passed in p_attributes using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.GET_USER_ATTRIBUTES(
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2,
    p_host              IN VARCHAR2,
    p_port              IN VARCHAR2 DEFAULT 389,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
```

```
    p_attributes       IN  apex_application_global.vc_arr2,
    p_attribute_values OUT apex_application_global.vc_arr2);
```

Parameters

Table 17-3 GET_USER_ATTRIBUTES Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).
p_attributes	An array of attribute names for which values are to be returned.
p_attribute_values	An array of values returned for each corresponding attribute name in p_attributes.

Example

The following example demonstrates how to use the APEX_LDAP.GET_USER_ATTRIBUTES procedure to retrieve a specific attribute value associated to a user.

```
DECLARE
    L_ATTRIBUTES apex_application_global.vc_arr2;
    L_ATTRIBUTE_VALUES apex_application_global.vc_arr2;
BEGIN
    L_ATTRIBUTES(1) := 'xxxxxxxxxx'; /* name of the employee number attribute */
    APEX_LDAP.GET_USER_ATTRIBUTES(
        p_username => 'firstname.lastname',
        p_pass => NULL,
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => '389',
        p_attributes => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);
END;
```

17.4 IS_MEMBER Function

The IS_MEMBER function returns a boolean TRUE if the user named by p_username (with password if required) is a member of the group specified by the p_group and p_group_base parameters using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.IS_MEMBER(
    p_username     IN VARCHAR2,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N',
    p_group        IN VARCHAR2,
    p_group_base   IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 17-4 IS_MEMBER Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL.
p_group	Name of the group to be search for membership.
p_group_base	The base from which the search should be started.

Example

The following example demonstrates how to use the APEX_LDAP.IS_MEMBER function to verify whether a user is a member of a group against an LDAP server.

```
DECLARE
    L_VAL boolean;
BEGIN
    L_VAL := APEX_LDAP.IS_MEMBER(
        p_username =>'firstname.lastname',
        p_pass =>'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => 389,
        p_group => 'group_name',
        p_group_base => 'group_base');
    IF L_VAL THEN
        htp.p('Is a member.');
    ELSE
        htp.p('Not a member.');
```

```
    END IF;
END;
```

17.5 MEMBER_OF Function

The MEMBER_OF function returns an array of groups the user name designated by p_username (with password if required) belongs to, using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.MEMBER_OF(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN apex_application_global.vc_arr2;
```

Parameters

Table 17-5 MEMBER_OF Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).

Example

The following example demonstrates how to use the APEX_LDAP.MEMBER_OF function to retrieve all the groups designated by the specified username.

```
DECLARE
    L_MEMBERSHIP       apex_application_global.vc_arr2;
BEGIN
    L_MEMBERSHIP := APEX_LDAP.MEMBER_OF(
        p_username         => 'firstname.lastname',
        p_pass             => 'abcdef',
        p_auth_base        => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host             => 'our_ldap_sever.my_company.com',
        p_port             => '389');
    FOR i IN L_MEMBERSHIP.FIRST..L_MEMBERSHIP.LAST LOOP
        htp.p('Member of: '||L_MEMBERSHIP(i));
    END LOOP;
END;
```

17.6 MEMBER_OF2 Function

The MEMBER_OF2 function returns a VARCHAR2 colon delimited list of groups the user name designated by p_username (with password if required) belongs to, using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.MEMBER_OF2(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN VARCHAR2;
```

Parameters

Table 17-6 MEMBER_OF2 Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).

Example

The following example demonstrates how to use the APEX_LDAP.MEMBER_OF2 function to retreive all the groups designated by the specified username.

```
DECLARE
    L_VAL varchar2(4000);
BEGIN
    L_VAL := APEX_LDAP.MEMBER_OF2(
        p_username => 'firstname.lastname',
        p_pass => 'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => 389);
    htp.p('Is Member of:'||L_VAL);
END;
```

17.7 SEARCH Function

The SEARCH function searches the LDAP repository. The result is an object table of (dn, name, val) that can be used in table queries.

Syntax

```
function search (
          p_username          IN VARCHAR2 DEFAULT NULL,
          p_pass              IN VARCHAR2 DEFAULT NULL,
          p_auth_base         IN VARCHAR2 DEFAULT NULL,
          p_host              IN VARCHAR2,
          p_port              IN NUMBER DEFAULT 389,
          p_use_ssl           IN VARCHAR2 DEFAULT 'N',
          p_search_base       IN VARCHAR2,
          p_search_filter     IN VARCHAR2,
          p_scope             IN BINARY_INTEGER DEFAULT SYS.DBMS_LDAP.SCOPE_SUBTREE,
          p_timeout_sec       IN BINARY_INTEGER DEFAULT 3,
          p_attribute_names   IN VARCHAR2 )
          RETURN APEX_T_LDAP_ATTRIBUTES PIPELINED;
```

Parameters

Table 17-7 Search Parameters

Parameter	Descriptions
p_username	Login name of the user (can be NULL for anonymous binds).
p_pass	The password for p_username (can be NULL for anonymous binds)
p_auth_base	The authentication base dn for p_username (for example, dc=users,dc=my,dc=org). Can be NULL for anonymous binds.
p_host	The LDAP server host name.
p_port	The LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).
p_search_base	dn base for the search.
p_search_filter	LDAP search filter expression.
p_scope	Search scope (default descends into subtrees).
p_timeout_sec	Timeout for the search (default is 3 seconds)
p_attribute_names	Comma separated list of return attribute names

Example 1

```
SELECT val group_dns
  FROM table(apex_ldap.search (
           p_host            => 'ldap.example.com',
           p_search_base     => 'dc=example,dc=com',
           p_search_filter   => 'uid='||apex_escape.ldap_search_filter(:APP_USER),
           p_attribute_names => 'memberof' ));
```

Example 2

```
SELECT dn, mail, dispname, phone
  from ( select dn, name, val
           from table(apex_ldap.search (
                          p_host            => 'ldap.example.com',
                          p_search_base     => 'dc=example,dc=com',
                          p_search_filter   => '&(objectClass=person)(ou=Test)',
                          p_attribute_names => 'mail,displayname,telephonenumber' )))
  pivot (min(val) for name in ( 'mail'            mail,
                                'displayname'     dispname,
                                'telephonenumber' phone ))
```

18

APEX_MAIL

You can use the APEX_MAIL package to send an email from an Oracle Application Express application. This package is built on top of the Oracle supplied UTL_SMTP package. Because of this dependence, the UTL_SMTP package must be installed and functioning to use APEX_MAIL.

APEX_MAIL contains three procedures. Use APEX_MAIL.SEND to send an outbound email message from your application. Use APEX_MAIL.PUSH_QUEUE to deliver mail messages stored in APEX_MAIL_QUEUE. Use APEX_MAIL.ADD_ATTACHMENT to send an outbound email message from your application as an attachment.

Note:

Oracle Application Express installs the database job ORACLE_APEX_MAIL_QUEUE, which periodically sends all mail messages stored in the active mail queue. In order to call the APEX_MAIL package from outside the context of an Application Express application, you must call apex_util.set_security_group_id as in the following example:

```
for c1 in (
   select workspace_id
     from apex_applications
    where application_id = p_app_id )
loop
   apex_util.set_security_group_id(p_security_group_id =>
c1.workspace_id);
end loop;
```

See Also:

- "*Oracle Database PL/SQL Packages and Types Reference*" for more information about the UTL_SMTP package
- "Sending Email from an Application" in *Oracle Application Express App Builder User's Guide*

18.1 Configuring Oracle Application Express to Send Email

Before you can send email from an App Builder application, you must:

1. Log in to Oracle Application Express Administration Services and configure the email settings on the Instance Settings page. See "Configuring Email" in *Oracle Application Express Administration Guide* .
2. If you are running Oracle Application Express with Oracle Database 11*g* release 1 (11.1), you must enable outbound mail. In Oracle Database 11*g* release 1 (11.1), the ability to interact with network services is disabled by default. See "Enabling Network Service in Oracle Database 11*g*" in*Oracle Application Express App Builder User's Guide*.

   Tip:

   You can configure Oracle Application Express to automatically email users their login credentials when a new workspace request has been approved. To learn more, see "Selecting a Provisioning Mode"in *Oracle Application Express Administration Guide*.

18.2 ADD_ATTACHMENT Procedure

This procedure sends an outbound email message from an application as an attachment. To add multiple attachments to a single email, `APEX_MAIL.ADD_ATTACHMENT` can be called repeatedly for a single email message.

Syntax

```
APEX_MAIL.ADD_ATTACHMENT(
    p_mail_id                   IN    NUMBER,
    p_attachment                IN    BLOB,
    p_filename                  IN    VARCHAR2,
    p_mime_type                 IN    VARCHAR2);
```

Parameters

Table 18-1 ADD_ATTACHMENT Parameters

Parameter	Description
`p_mail_id`	The numeric ID associated with the email. This is the numeric identifier returned from the call to `APEX_MAIL.SEND` to compose the email body.
`p_attachment`	A `BLOB` variable containing the binary content to be attached to the email message.

Table 18-1 (Cont.) ADD_ATTACHMENT Parameters

Parameter	Description
p_filename	The filename associated with the email attachment.
p_mime_type	A valid MIME type (or Internet media type) to associate with the email attachment.

Examples

The following example demonstrates how to access files stored in APEX_APPLICATION_FILES and add them to an outbound email message

```
DECLARE
    l_id NUMBER;
BEGIN
    l_id := APEX_MAIL.SEND(
        p_to        => 'fred@flintstone.com',
        p_from      => 'barney@rubble.com',
        p_subj      => 'APEX_MAIL with attachment',
        p_body      => 'Please review the attachment.',
        p_body_html => '<b>Please</b> review the attachment');
    FOR c1 IN (SELECT filename, blob_content, mime_type
        FROM APEX_APPLICATION_FILES
        WHERE ID IN (123,456)) LOOP

        APEX_MAIL.ADD_ATTACHMENT(
            p_mail_id    => l_id,
            p_attachment => c1.blob_content,
            p_filename   => c1.filename,
            p_mime_type  => c1.mime_type);
        END LOOP;
    COMMIT;
END;
/
```

18.3 GET_IMAGES_URL Function

Use this function to get the image prefixed URL, if the email includes Application Express instance images.

Syntax

```
APEX_MAIL.GET_IMAGES_URL return VARCHAR2;
```

Parameters

None.

Example

The following example sends an Order Confirmation email which includes the Oracle Logo image.

```
declare
    l_body      clob;
    l_body_html clob;
```

```
begin
    l_body := 'To view the content of this message, please use an HTML enabled mail
client.' || utl_tcp.crlf;

    l_body_html := '<html><body>' || utl_tcp.crlf ||
                   '<p>Please confirm your order on the <a href="' ||
                   apex_mail.get_instance_url || 'f?p=100:10">Order Confirmation</a>
page.</p>' || utl_tcp.crlf ||
                   '<p>Sincerely,<br />' || utl_tcp.crlf ||
                   'The Application Express Dev Team<br />' || utl_tcp.crlf ||
                   '<img src="' || apex_mail.get_images_url || 'oracle.gif"
alt="Oracle Logo"></p>' || utl_tcp.crlf ||
                   '</body></html>';
    apex_mail.send (
        p_to        => 'some_user@somewhere.com',   -- change to your email address
        p_from      => 'some_sender@somewhere.com', -- change to a real senders
email address
        p_body      => l_body,
        p_body_html => l_body_html,
        p_subj      => 'Order Confirmation' );
end;
```

18.4 GET_INSTANCE_URL Function

If an email includes a link to an Application Express instance, use this function to get the instance URL.

Note:

This function requires that the instance setting `Application Express Instance URL` for emails is set.

Syntax

```
APEX_MAIL.GET_INSTANCE_URL return VARCHAR2;
```

Parameters

None.

Example

The following example sends an Order Confirmation email which includes an absolute URL to page 10 of application 100.

```
declare
    l_body      clob;
    l_body_html clob;
begin
    l_body := 'To view the content of this message, please use an HTML enabled mail
client.' || utl_tcp.crlf;

    l_body_html := '<html><body>' || utl_tcp.crlf ||
                   '<p>Please confirm your order on the <a href="' ||
                   apex_mail.get_instance_url || 'f?p=100:10">Order Confirmation</a>
page.</p>' || utl_tcp.crlf ||
                   '</body></html>';
    apex_mail.send (
```

```
        p_to        => 'some_user@somewhere.com',   -- change to your email address
        p_from      => 'some_sender@somewhere.com', -- change to a real senders
email address
        p_body      => l_body,
        p_body_html => l_body_html,
        p_subj      => 'Order Confirmation' );
end;
```

18.5 PUSH_QUEUE Procedure

Oracle Application Express stores unsent email messages in a table named APEX_MAIL_QUEUE. You can manually deliver mail messages stored in this queue to the specified SMTP gateway by invoking the APEX_MAIL.PUSH_QUEUE procedure.

Oracle Application Express logs successfully submitted message in the table APEX_MAIL_LOG with the timestamp reflecting your server's local time.

Syntax

```
APEX_MAIL.PUSH_QUEUE(
    p_smtp_hostname             IN    VARCHAR2 DEFAULT NULL,
    p_smtp_portno               IN    NUMBER   DEFAULT NULL);
```

Parameters

Table 18-2 PUSH_QUEUE Parameters

Parameters	Description
p_smtp_hostname	SMTP gateway host name
p_smtp_portno	SMTP gateway port number

Note that these parameter values are provided for backward compatibility, but their respective values are ignored. The SMTP gateway hostname and SMTP gateway port number are exclusively derived from values entered on the Manage Environment Settings when sending email.

Example

The following example demonstrates the use of the APEX_MAIL.PUSH_QUEUE procedure using a shell script. This example only applies to UNIX/LINUX installations.

```
SQLPLUS / <<EOF
APEX_MAIL.PUSH_QUEUE;
DISCONNECT
EXIT
EOF
```

See Also:

- "Sending an Email from an Application" in *Oracle Application Express App Builder User's Guide*
- "Configuring Email Settings" in *Oracle Application Express Administration Guide*
- "Sending Email from an Application" in *Oracle Application Express App Builder User's Guide*

18.6 SEND Procedure

This procedure sends an outbound email message from an application. Although you can use this procedure to pass in either a `VARCHAR2` or a `CLOB` to `p_body` and `p_body_html`, the data types must be the same. In other words, you cannot pass a `CLOB` to `P_BODY` and a `VARCHAR2` to `p_body_html`.

When using `APEX_MAIL.SEND`, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your `p_body` or `p_body_html` parameters into chunks of 1000 characters or less. Failing to do so results in erroneous email messages, including partial messages or messages with extraneous exclamation points.
- **Plain text and HTML email content.** Passing a value to `p_body`, but not `p_body_html` results in a plain text message. Passing a value to `p_body` and `p_body_html` yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient's email client determine what displays. Although most modern email clients can read an HTML formatted email, remember that some users disable this functionality to address security issues.
- **Avoid images.** When referencing images in `p_body_html` using the `<img />` tag, remember that the images must be accessible to the recipient's email client in order for them to see the image.

  For example, suppose you reference an image on your network called `hello.gif` as follows:

  ```
  <img src="http://someserver.com/hello.gif" alt="Hello" />]
  ```

  In this example, the image is not attached to the email, but is referenced by the email. For the recipient to see it, they must be able to access the image using a web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image is not displayed. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```
APEX_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
```

```
p_body_html                IN  [ VARCHAR2 | CLOB ] DEFAULT NULL,
p_subj                     IN    VARCHAR2 DEFAULT NULL,
p_cc                       IN    VARCHAR2 DEFAULT NULL,
p_bcc                      IN    VARCHAR2 DEFAULT NULL,
p_replyto                  IN    VARCHAR2);
```

Parameters

Table 18-3 SEND Parameters

Parameter	Description
p_to	Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list
p_from	Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent
p_body	Body of the email in plain text, not HTML (required). If a value is passed to p_body_html, then this is the only text the recipient sees. If a value is not passed to p_body_html, then this text only displays for email clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters.
p_body_html	Body of the email in HTML format. This must be a full HTML document including the <html> and <body> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)
p_subj	Subject of the email
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list
p_replyto	Address of the Reply-To mail header. You can use this parameter as follows: • If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter • If you include the p_replyto parameter, but provide a NULL value, the Reply-To mail header is set to NULL. This results in the suppression of automatic email replies • If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you send these messages, but the automatic replies go to the value specified (for example, the email address)

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
BEGIN
    l_body := 'Thank you for your interest in the APEX_MAIL
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||'  Sincerely,'||utl_tcp.crlf;
    l_body := l_body ||'  The Application Express Dev Team'||utl_tcp.crlf;
    apex_mail.send(
        p_to       => 'some_user@somewhere.com',   -- change to your email address
        p_from     => 'some_sender@somewhere.com', -- change to a real senders email
address
        p_body     => l_body,
        p_subj     => 'APEX_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use `APEX_MAIL.SEND` to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses `utl_tcp.crlf`.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
BEGIN
    l_body := 'To view the content of this message, please use an HTML enabled mail
client.'||utl_tcp.crlf;

    l_body_html := '<html>
        <head>
            <style type="text/css">
                body{font-family: Arial, Helvetica, sans-serif;
                    font-size:10pt;
                    margin:30px;
                    background-color:#ffffff;}

                span.sig{font-style:italic;
                    font-weight:bold;
                    color:#811919;}
             </style>
         </head>
         <body>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'<p>Thank you for your interest in the
<strong>APEX_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  <span class="sig">The Application Express Dev
Team</span><br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'</body></html>';
    apex_mail.send(
    p_to   => 'some_user@somewhere.com',   -- change to your email address
    p_from => 'some_sender@somewhere.com', -- change to a real senders email address
    p_body      => l_body,
    p_body_html => l_body_html,
    p_subj      => 'APEX_MAIL Package - HTML formatted message');
END;
/
```

18.7 SEND Function

This function sends an outbound email message from an application. Although you can use this function to pass in either a VARCHAR2 or a CLOB to p_body and p_body_html, the data types must be the same. In other words, you cannot pass a CLOB to P_BODY and a VARCHAR2 to p_body_html.

This function returns a NUMBER. The NUMBER returned is the unique numeric identifier associated with the mail message.

When using APEX_MAIL.SEND, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your p_body or p_body_html parameters into chunks of 1000 characters or less. Failing to do so results in erroneous email messages, including partial messages or messages with extraneous exclamation points.
- **Plain text and HTML email content.** Passing a value to p_body, but not p_body_html results in a plain text message. Passing a value to p_body and p_body_html yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient's email client determine what displays. Although most modern email clients can read an HTML formatted email, remember that some users disable this functionality to address security issues.
- **Avoid images.** When referencing images in p_body_html using the <img /> tag, remember that the images must be accessible to the recipient's email client in order for them to see the image.

  For example, suppose you reference an image on your network called hello.gif as follows:

  ```
  <img src="http://someserver.com/hello.gif" alt="Hello" />]
  ```

  In this example, the image is not attached to the email, but is referenced by the email. For the recipient to see it, they must be able to access the image using a web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image is not displayed. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```
APEX_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
    p_body_html                 IN  [ VARCHAR2 | CLOB ] DEFAULT NULL,
    p_subj                      IN    VARCHAR2 DEFAULT NULL,
    p_cc                        IN    VARCHAR2 DEFAULT NULL,
    p_bcc                       IN    VARCHAR2 DEFAULT NULL,
    p_replyto                   IN    VARCHAR2)
return NUMBER;
```

Parameters

Table 18-4 SEND Parameters

Parameter	Description
p_to	Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list
p_from	Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent
p_body	Body of the email in plain text, not HTML (required). If a value is passed to p_body_html, then this is the only text the recipient sees. If a value is not passed to p_body_html, then this text only displays for email clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters.
p_body_html	Body of the email in HTML format. This must be a full HTML document including the <html> and <body> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)
p_subj	Subject of the email
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list
p_replyto	Address of the Reply-To mail header. You can use this parameter as follows: • If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter • If you include the p_replyto parameter, but provide a NULL value, the Reply-To mail header is set to NULL. This results in the suppression of automatic email replies • If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you send these messages, but the automatic replies go to the value specified (for example, the email address)

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application and return the unique message ID.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
    l_id NUMBER;
BEGIN
    l_body := 'Thank you for your interest in the APEX_MAIL
```

```
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||'  Sincerely,'||utl_tcp.crlf;
    l_body := l_body ||'  The Application Express Dev Team'||utl_tcp.crlf;
    l_id   := apex_mail.send(
        p_to       => 'some_user@somewhere.com',   -- change to your email address
        p_from     => 'some_sender@somewhere.com', -- change to a real senders email
address
        p_body     => l_body,
        p_subj     => 'APEX_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use `APEX_MAIL.SEND` to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses `utl_tcp.crlf`.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
    l_id NUMBER;
BEGIN
    l_body := 'To view the content of this message, please use an HTML enabled mail
client.'||utl_tcp.crlf;

    l_body_html := '<html>
        <head>
            <style type="text/css">
                body{font-family: Arial, Helvetica, sans-serif;
                    font-size:10pt;
                    margin:30px;
                    background-color:#ffffff;}

                span.sig{font-style:italic;
                    font-weight:bold;
                    color:#811919;}
             </style>
         </head>
         <body>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'<p>Thank you for your interest in the
<strong>APEX_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  <span class="sig">The Application Express Dev
Team</span><br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'</body></html>';
    l_id        := apex_mail.send(
        p_to        => 'some_user@somewhere.com',   -- change to your email address
        p_from      => 'some_sender@somewhere.com', -- change to a real senders
email address
        p_body      => l_body,
        p_body_html => l_body_html,
        p_subj      => 'APEX_MAIL Package - HTML formatted message');
END;
/
```

19

APEX_PAGE

The APEX_PAGE package is the public API for handling pages.

19.1 Global Constants

The following constants are used by this package.

```
c_ui_type_desktop        constant varchar2(10) := 'DESKTOP';
c_ui_type_jqm_smartphone constant varchar2(15) := 'JQM_SMARTPHONE';
```

19.2 IS_DESKTOP_UI Function

This function returns TRUE if the current page has been designed for desktop browsers.

Syntax

```
FUNCTION IS_DESKTOP_UI
RETURN BOOLEAN;
```

19.3 IS_JQM_SMARTPHONE_UI Function

This function returns TRUE if the current page has been designed for smartphone devices using jQuery Mobile.

Syntax

```
FUNCTION IS_JQM_SMARTPHONE_UI
RETURN BOOLEAN;
```

19.4 IS_JQM_TABLET_UI Function

This function returns TRUE if the current page has been designed for tablet devices using jQuery Mobile.

Syntax

```
FUNCTION IS_JQM_TABLET_UI
RETURN BOOLEAN;
```

19.5 GET_UI_TYPE Function

This function returns the user interface (UI) type for which the current page has been designed.

Syntax

```
FUNCTION GET_UI_TYPE
RETURN VARCHAR2;
```

19.6 IS_READ_ONLY Function

This function returns TRUE if the current page is rendered read-only and FALSE if it is not.

Syntax

```
FUNCTION IS_READ_ONLY
RETURN BOOLEAN;
```

19.7 GET_PAGE_MODE Function

This function returns the page mode for the current page.

Syntax

```
FUNCTION GET_PAGE_MODE (
    p_application_id        IN NUMBER,
    p_page_id
    RETURN VARCHAR2;
```

Parameters

Table 19-1 GET_PAGE_MODE Parameters

Parameter	Description
p_application_id	ID of the application. Defaults to the current application.
p_page_id	ID of the page. Defaults to the current page.

19.8 PURGE_CACHE Procedure

This procedure purges the cache of the specified application, page, and region for the specified user. If the user is not specified, the procedure purges all cached versions of the page.

Syntax

```
PROCEDURE PURGE_CACHE (
    p_application_id       IN NUMBER DEFAULT apex.g_flow_id,
    p_page_id              IN NUMBER DEFAULT apex.g_flow_step_id,
    p_user_name            IN VARCHAR2 DEFAULT NULL,
    p_current_session_only IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Table 19-2 PURGE_CACHE Parameters

Parameter	Description
p_application_id	ID of the application. Defaults to the current application.
p_page_id	ID of the page.Defaults to the current page. If you pass NULL, Oracle Application Express purges the cache on all pages of the application.
p_user_name	Specify a user name if you only want to purge entries that were saved for the given user.
p_current_session_only	Specify TRUE if you only want to purge entries that where saved for the current session. Defaults to FALSE.

Example

This example purges session specific cache on the current page.

```
BEGIN
     APEX_PAGE.PURGE_CACHE (
         p_current_session_only => true );
END;
```

19.9 GET_URL Function

This function returns an Oracle Application Express f?p= URL. It is sometimes clearer to read a function call than a concatenated URL. See the example below for a comparison.

Syntax

```
FUNCTION GET_URL (
    p_application        IN VARCHAR2 DEFAULT NULL,
    p_page               IN VARCHAR2 DEFAULT NULL,
    p_session            IN NUMBER   DEFAULT APEX.G_INSTANCE,
    p_request            IN VARCHAR2 DEFAULT NULL,
    p_debug              IN VARCHAR2 DEFAULT NULL,
    p_clear_cache        IN VARCHAR2 DEFAULT NULL,
    p_items              IN VARCHAR2 DEFAULT NULL,
    p_values             IN VARCHAR2 DEFAULT NULL,
```

```
    p_printer_friendly   IN VARCHAR2 DEFAULT NULL,
    p_trace              IN VARCHAR2 DEFAULT NULL )
    RETURN VARCHAR2;
```

Parameters

Table 19-3 GET_URL Parameters

Parameter	Description
p_application	The application ID or alias. Defaults to the current application.
p_page	Page ID or alias. Defaults to the current page.
p_session	Session ID. Defaults to the current session ID.
p_request	URL request parameter.
p_debug	URL debug parameter. Defaults to the current debug mode.
p_clear_cache	URL clear cache parameter.
p_items	Comma-delimited list of item names to set session state.
p_values	Comma-delimited list of item values to set session state.
p_printer_friendly	URL printer friendly parameter. Defaults to the current request's printer friendly mode.
p_trace	SQL trace parameter.

Example

This query uses APEX_PAGE.GET_URL and its alternative APEX_UTIL.PREPARE_URL to produce two identical URLs.

```
SELECT APEX_PAGE.GET_URL (
            p_page   => 1,
            p_items  => 'P1_X,P1_Y',
            p_values => 'somevalue,othervalue' ) f_url_1,
         APEX_UTIL.PREPARE_URL('f?p=&APP_ID.:
1:&APP_SESSION.::::P1_X,P1_Y:somevalue,othervalue')
     FROM DUAL
```

20

APEX_PLUGIN

The APEX_PLUGIN package provides the interface declarations and some utility functions to work with plug-ins.

20.1 Data Types

The data types used by the APEX_PLUGIN package are described in this section.

Data Types:

- t_process (page 20-6)
- t_process_exec_result (page 20-7)
- type t_region_column ((page 20-7)
- type t_region_columns is table of t_region_column index by pls_integer; (page 20-8)
- type t_region_column ((page 20-7)
- type t_region_columns is table of t_region_column index by pls_integer; (page 20-8)
- t_region (page 20-8)
- t_region_ajax_result (page 20-8)
- t_region_render_result (page 20-8)

c_*

The following constants are used for `display_location` in the page item validation function result type `t_page_item_validation_result`.

```
c_inline_with_field           constant varchar2(40) := 'INLINE_WITH_FIELD';
c_inline_with_field_and_notif constant varchar2(40) :=
'INLINE_WITH_FIELD_AND_NOTIFICATION';
c_inline_in_notification      constant varchar2(40) := 'INLINE_IN_NOTIFICATION';
c_on_error_page               constant varchar2(40) := 'ON_ERROR_PAGE';
```

t_authentication

```
  type t_authentication is record (
    id                   number,
    name                 varchar2(255),
    invalid_session_url  varchar2(4000),
    logout_url           varchar2(4000),
    plsql_code           clob,
    attribute_01         varchar2(32767),
    attribute_02         varchar2(32767),
    attribute_03         varchar2(32767),
    attribute_04         varchar2(32767),
    attribute_05         varchar2(32767),
    attribute_06         varchar2(32767),
    attribute_07         varchar2(32767),
    attribute_08         varchar2(32767),
    attribute_09         varchar2(32767),
    attribute_10         varchar2(32767),
    attribute_11         varchar2(32767),
    attribute_12         varchar2(32767),
    attribute_13         varchar2(32767),
    attribute_14         varchar2(32767),
    attribute_15         varchar2(32767),
    --
    session_id           number,
    username             varchar2(255) );
```

t_authentication_ajax_result

```
type t_authentication_ajax_result is record (
  dummy                boolean );
```

t_authentication_auth_result

```
type t_authentication_auth_result is record (
  is_authenticated     boolean,
  redirect_url         varchar2(4000),
  log_code             number,
  log_text             varchar2(4000),
  display_text         varchar2(4000) );
```

t_authentication_inval_result

```
type t_authentication_inval_result is record (
  redirect_url         varchar2(4000) );
```

t_authentication_logout_result

```
type t_authentication_logout_result is record (
  redirect_url         varchar2(4000) );
```

t_authentication_sentry_result

```
type t_authentication_sentry_result is record (
  is_valid             boolean );
```

t_authorization

The following type is passed to all authorization plug-in functions and contains information about the current authorization.

```
type t_authorization is record (
  id                   number,
  name                 varchar2(255),
  username             varchar2(255),
  caching              varchar2(20),
  component            apex.t_component,
  attribute_01         varchar2(32767),
  attribute_02         varchar2(32767),
  attribute_03         varchar2(32767),
  attribute_04         varchar2(32767),
  attribute_05         varchar2(32767),
  attribute_06         varchar2(32767),
  attribute_07         varchar2(32767),
  attribute_08         varchar2(32767),
  attribute_09         varchar2(32767),
  attribute_10         varchar2(32767),
  attribute_11         varchar2(32767),
  attribute_12         varchar2(32767),
  attribute_13         varchar2(32767),
  attribute_14         varchar2(32767),
  attribute_15         varchar2(32767),
```

t_authorization_exec_result

The `t_authorization_exec_result` data type has been added to the `APEX_PLUGIN` package.

```
  type t_authorization_exec_result is record (
    is_authorized        boolean
    );
```

t_dynamic_action

The following type is passed into all dynamic action plug-in functions and contains information about the current dynamic action.

```
type t_dynamic_action is record (
    id           number,
    action       varchar2(50),
    attribute_01 varchar2(32767),
    attribute_02 varchar2(32767),
    attribute_03 varchar2(32767),
    attribute_04 varchar2(32767),
    attribute_05 varchar2(32767),
    attribute_06 varchar2(32767),
    attribute_07 varchar2(32767),
    attribute_08 varchar2(32767),
    attribute_09 varchar2(32767),
    attribute_10 varchar2(32767),
    attribute_11 varchar2(32767),
    attribute_12 varchar2(32767),
    attribute_13 varchar2(32767),
    attribute_14 varchar2(32767),
    attribute_15 varchar2(32767) );
```

t_dynamic_action_ajax_result

The following type is used as the result type for the Ajax function of a dynamic action type plug-in.

```
type t_dynamic_action_ajax_result is record (
    dummy boolean /* not used yet */
    );
```

t_dynamic_action_render_result

The following type is used as the result type for the rendering function of a dynamic action plug-in.

```
type t_dynamic_action_render_result is record (
    javascript_function varchar2(32767),
    ajax_identifier     varchar2(255),
    attribute_01        varchar2(32767),
    attribute_02        varchar2(32767),
    attribute_03        varchar2(32767),
    attribute_04        varchar2(32767),
    attribute_05        varchar2(32767),
    attribute_06        varchar2(32767),
    attribute_07        varchar2(32767),
    attribute_08        varchar2(32767),
    attribute_09        varchar2(32767),
    attribute_10        varchar2(32767),
    attribute_11        varchar2(32767),
    attribute_12        varchar2(32767),
    attribute_13        varchar2(32767),
    attribute_14        varchar2(32767),
    attribute_15        varchar2(32767) );
```

t_page_item

The following type is passed into all item type plug-in functions and contains information about the current page item.

```
type t_page_item is record (
    id                          number,
    name                        varchar2(255),
    label                       varchar2(4000),
    plain_label                 varchar2(4000),
    label_id                    varchar2(255), /* label id is set if 'Standard Form
Element' = no and label template uses #LABEL_ID# substitution */
    placeholder                 varchar2(255),
    format_mask                 varchar2(255),
    is_required                 boolean,
    lov_definition              varchar2(4000),
    lov_display_extra           boolean,
    lov_display_null            boolean,
    lov_null_text               varchar2(255),
    lov_null_value              varchar2(255),
    lov_cascade_parent_items    varchar2(255),
    ajax_items_to_submit        varchar2(255),
    ajax_optimize_refresh       boolean,
    element_width               number,
    element_max_length          number,
    element_height              number,
    element_css_classes         varchar2(255),
    element_attributes          varchar2(2000),
    element_option_attributes   varchar2(4000),
    escape_output               boolean,
    attribute_01                varchar2(32767),
    attribute_02                varchar2(32767),
    attribute_03                varchar2(32767),
    attribute_04                varchar2(32767),
    attribute_05                varchar2(32767),
    attribute_06                varchar2(32767),
    attribute_07                varchar2(32767),
    attribute_08                varchar2(32767),
    attribute_09                varchar2(32767),
    attribute_10                varchar2(32767),
    attribute_11                varchar2(32767),
    attribute_12                varchar2(32767),
    attribute_13                varchar2(32767),
    attribute_14                varchar2(32767),
    attribute_15                varchar2(32767) );
```

t_page_item_ajax_result

The following type is used as the result type for the Ajax function of an item type plug-in.

```
type t_page_item_ajax_result is record (
    dummy boolean /* not used yet */
    );
```

t_page_item_render_result

The following type is used as the result type for the rendering function of an item type plug-in.

```
type t_page_item_render_result is record (
    is_navigable     boolean default false,
    navigable_dom_id varchar2(255)          /* should only be set if navigable
element is not equal to item name */
    );
```

t_page_item_validation_result

The following type is used as the result type for the validation function of an item type plug-in.

```
type t_page_item_validation_result is record (
    message          varchar2(32767),
    display_location varchar2(40),    /* if not set the application default is used
*/
    page_item_name   varchar2(255) ); /* if not set the validated page item name is
used */
```

t_plugin

The following type is passed into all plug-in functions and contains information about the current plug-in.

```
type t_plugin is record (
    name         varchar2(45),
    file_prefix  varchar2(4000),
    attribute_01 varchar2(32767),
    attribute_02 varchar2(32767),
    attribute_03 varchar2(32767),
    attribute_04 varchar2(32767),
    attribute_05 varchar2(32767),
    attribute_06 varchar2(32767),
    attribute_07 varchar2(32767),
    attribute_08 varchar2(32767),
    attribute_09 varchar2(32767),
    attribute_10 varchar2(32767),
    attribute_11 varchar2(32767),
    attribute_12 varchar2(32767),
    attribute_13 varchar2(32767),
    attribute_14 varchar2(32767),
    attribute_15 varchar2(32767) );
```

t_process

The following type is passed into all process type plug-in functions and contains information about the current process.

```
type t_process is record (
    id                  number,
    name                varchar2(255),
    success_message     varchar2(32767),
    attribute_01        varchar2(32767),
    attribute_02        varchar2(32767),
    attribute_03        varchar2(32767),
    attribute_04        varchar2(32767),
    attribute_05        varchar2(32767),
    attribute_06        varchar2(32767),
    attribute_07        varchar2(32767),
    attribute_08        varchar2(32767),
    attribute_09        varchar2(32767),
    attribute_10        varchar2(32767),
```

```
    attribute_11        varchar2(32767),
    attribute_12        varchar2(32767),
    attribute_13        varchar2(32767),
    attribute_14        varchar2(32767),
    attribute_15        varchar2(32767) );
```

t_process_exec_result

The following type is used as the result type for the execution function of a process type plug-in.

```
type t_process_exec_result is record (
    success_message varchar2(32767)
    execution_skipped boolean default false /* set to TRUE if process execution has
been skipped by plug-in because of additional condition checks */
    );
```

type t_region_column (

The following type is passed into all region type plug-in functions and contains information about the current region.

```
type  t_region_column is record (
    id                  number,
    name                t_region_column_name,
    is_displayed        boolean,
    heading             apex_region_columns.heading%type,
    heading_alignment   apex_region_columns.heading_alignment%type,
    value_alignment     apex_region_columns.value_alignment%type,
    value_css_classes   apex_region_columns.value_css_classes%type,
    value_attributes    apex_region_columns.value_attributes%type,
    format_mask         apex_region_columns.format_mask%type,
    escape_output       boolean,
    attribute_01        varchar2(32767),
    attribute_02        varchar2(32767),
    attribute_03        varchar2(32767),
    attribute_04        varchar2(32767),
    attribute_05        varchar2(32767),
    attribute_06        varchar2(32767),
    attribute_07        varchar2(32767),
    attribute_08        varchar2(32767),
    attribute_09        varchar2(32767),
    attribute_10        varchar2(32767),
    attribute_11        varchar2(32767),
    attribute_12        varchar2(32767),
    attribute_13        varchar2(32767),
    attribute_14        varchar2(32767),
    attribute_15        varchar2(32767),
    attribute_16        varchar2(32767),
    attribute_17        varchar2(32767),
    attribute_18        varchar2(32767),
    attribute_19        varchar2(32767),
    attribute_20        varchar2(32767),
    attribute_21        varchar2(32767),
    attribute_22        varchar2(32767),
    attribute_23        varchar2(32767),
    attribute_24        varchar2(32767),
    attribute_25        varchar2(32767);
```

type t_region_columns is table of t_region_column index by pls_integer;

t_region

The following type is passed into all region type plug-in functions and contains information about the current region.

```
type t_region is record (
    id                    number,
    static_id              varchar2(255),
    name                   varchar2(255),
    type                   varchar2(255),
    source                 varchar2(32767),
    ajax_items_to_submit   varchar2(32767),
    fetched_rows           pls_integer,
    escape_output          boolean,
    error_message          varchar2(32767), /* obsolete */
    no_data_found_message varchar2(32767),
    attribute_01          varchar2(32767),
    attribute_02          varchar2(32767),
    attribute_03          varchar2(32767),
    attribute_04          varchar2(32767),
    attribute_05          varchar2(32767),
    attribute_06          varchar2(32767),
    attribute_07          varchar2(32767),
    attribute_08          varchar2(32767),
    attribute_09          varchar2(32767),
    attribute_10          varchar2(32767),
    attribute_11          varchar2(32767),
    attribute_12          varchar2(32767),
    attribute_13          varchar2(32767),
    attribute_14          varchar2(32767),
    attribute_15          varchar2(32767),
    attribute_16          varchar2(32767),
    attribute_17          varchar2(32767),
    attribute_18          varchar2(32767),
    attribute_19          varchar2(32767),
    attribute_20          varchar2(32767),
    attribute_21          varchar2(32767),
    attribute_22          varchar2(32767),
    attribute_23          varchar2(32767),
    attribute_24          varchar2(32767),
    attribute_25          varchar2(32767),
    region_columns        t_region_columns );
```

t_region_ajax_result

The following type is used as result type for the Ajax function of a region type plug-in.

```
type t_region_ajax_result is record (
    dummy boolean /* not used yet */
    );
```

t_region_render_result

The following type is used as the result type for the rendering function of a region type plug-in.

```
type t_region_render_result is record (
    navigable_dom_id varchar2(255) /* can be used to put focus to an input field
```

```
(that is, search field) the region renders as part of the plug-in output */
    );
```

20.2 GET_AJAX_IDENTIFIER Function

This function returns the Ajax identifier used to call the Ajax callback function defined for the plug-in.

Note:

This function only works in the context of a plug-in rendering function call and only if the plug-in has defined an Ajax function callback in the plug-in definition.

Syntax

```
APEX_PLUGIN.GET_AJAX_IDENTIFIER
RETURN VARCHAR2;
```

Parameters

None.

Example

This is an example of a dynamic action plug-in rendering function that supports an Ajax callback.

```
function render_set_value (
    p_dynamic_action in apex_plugin.t_dynamic_action )
    return apex_plugin.t_dynamic_action_render_result
is
    l_result                apex_plugin.t_dynamic_action_render_result;
begin
    l_result.javascript_function := 'com_oracle_apex_set_value';
    l_result.ajax_identifier     := apex_plugin.get_ajax_identifier;
    return l_result;
end;
```

20.3 GET_INPUT_NAME_FOR_PAGE_ITEM Function

Use this function when you want to render an HTML input element in the rendering function of an item type plug-in.For the HTML input element, for example, `<input type="text" id="P1_TEST" name="xxx">`, you have to provide a value for the `name` attribute so that Oracle Application Express can map the submitted value to the actual page item in session state. This function returns the mapping `name` for your page item. If the HTML input element has multiple values, such as a select list with `multiple="multiple"`, then set `p_is_multi_value` to TRUE.

Note:

This function is only useful when called in the rendering function of an item type plug-in.

Syntax

```
APEX_PLUGIN.GET_INPUT_NAME_FOR_PAGE_ITEM (
    p_is_multi_value  IN BOOLEAN)
RETURN VARCHAR2;
```

Parameters

Table 20-1 GET_INPUT_NAME_FOR_PAGE_ITEM Parameters

Parameter	Description
p_is_multi_value	Set to TRUE if the HTML input element has multiple values. If not, set to FALSE. HTML input elements with multiple values can be checkboxes and multi select lists.

Example

The following example outputs the necessary HTML code to render a text field where the value gets stored in session state when the page is submitted.

```
sys.htp.prn (
    '<input type="text" id="'||p_item.name||'" '||
    'name="'||apex_plugin.get_input_name_for_page_item(false)||'" '||
    'value="'||sys.htf.escape_sc(p_value)||'" '||
    'size="'||p_item.element_width||'" '||
    'maxlength="'||p_item.element_max_length||'" '||
    coalesce(p_item.element_attributes, 'class="text_field"')||' />' );
```

21

APEX_PLUGIN_UTIL

The `APEX_PLUGIN_UTIL` package provides utility functions that solve common problems when writing a plug-in.

21.1 CLEAR_COMPONENT_VALUES Procedure

This procedure clears the component specific `Session State` set by `apex_plugin_util.set_component_values`.

Syntax

```
procedure clear_component_values;
```

Example

See `apex_plugin_util.set_component_values`

See Also:

SET_COMPONENT_VALUES Procedure (page 21-33)

21.2 DEBUG_DYNAMIC _ACTION Procedure

This procedure writes the data of the dynamic action meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_DYNAMIC_ACTION (
    p_plugin         IN apex_plugin.t_plugin,
    p_dynamic_action IN apex_plugin.t_dynamic_action);
```

Parameters

Table 21-1 DEBUG_DYNAMIC_ACTION Parameters

Parameter	Description
`p_plugin`	This is the `p_plugin` parameter of your plug-in function.
`p_dynamic_action`	This is the `p_dynamic_action` parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the rendered function or Ajax callback function of the plug-in.

```
apex_plugin_util.debug_dynamic_action (
    p_plugin         => p_plugin,
    p_dynamic_action => p_dynamic_action );
```

21.3 DEBUG_PAGE_ITEM Procedure Signature 1

This procedure writes the data of the page item meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_PAGE_ITEM (
    p_plugin    IN apex_plugin.t_plugin,
    p_page_item IN apex_plugin.t_page_item);
```

Parameters

Table 21-2 DEBUG_PAGE_ITEM Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_page_item	This is the p_page_item parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the renderer, Ajax callback or validation function.

```
apex_plugin_util.debug_page_item (
    p_plugin    => p_plugin,
    p_page_item => p_page_item );
```

21.4 DEBUG_PAGE_ITEM Procedure Signature 2

This procedure writes the data of the page item meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_PAGE_ITEM (
    p_plugin              IN apex_plugin.t_plugin,
    p_page_item           IN apex_plugin.t_page_item,
    p_value               IN VARCHAR2,
    p_is_readonly         IN BOOLEAN,
    p_is_printer_friendly IN BOOLEAN);
```

Parameters

Table 21-3 DEBUG_PAGE_ITEM Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_page_item	This is the p_page_item parameter of your plug-in function.
p_value	This is the p_value parameter of your plug-in function.
p_is_readonly	This is the p_is_readonly parameter of your plug-in function.
p_is_printer_friendly	This is the p_is_printer_friendly parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the renderer, Ajax callback or validation function.

```
apex_plugin_util.debug_page_item (
    p_plugin         => p_plugin,
    p_page_item      => p_page_item,
    p_value          => p_value,
    p_is_readonly    => p_is_readonly,
    p_is_printer_friendly => p_is_printer_friendly);
```

21.5 DEBUG_PROCESS Procedure

This procedure writes the data of the process meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_PROCESS (
    p_plugin         IN apex_plugin.t_plugin,
    p_process        IN apex_plugin.t_process);
```

Parameters

Table 21-4 DEBUG_PROCESS Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_process	This is the p_process parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the execution function of the plug-in.

```
apex_plugin_util.debug_process (
    p_plugin         => p_plugin,
    p_process        => p_process);
```

21.6 DEBUG_REGION Procedure Signature 1

This procedure writes the data of the region meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_REGION (
    p_plugin              IN apex_plugin.t_plugin,
    p_region              IN apex_plugin.t_region);
```

Parameters

Table 21-5 DEBUG_REGION Signature 1 Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_region	This is the p_region parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the render function or Ajax callback function of the plug-in.

```
apex_plugin_util.debug_process (
    p_plugin         => p_plugin,
    p_region        => p_region);
```

21.7 DEBUG_REGION Procedure Signature 2

This procedure writes the data of the region meta data to the debug output if debugging is enabled. This is the advanced version of the debugging procedure which is used for the rendering function of a region plug-in.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_REGION (
    p_plugin              IN apex_plugin.t_plugin,
    p_region              IN apex_plugin.t_region,
    p_is_printer_friendly IN BOOLEAN);
```

Parameters

Table 21-6 (page 21-6) describes the parameters available in the DEBUG_REGION procedure.

Table 21-6 DEBUG_REGION Signature 2 Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function
p_region	This is the p_region parameter of your plug-in function
p_is_printer_friendly	This is the p_is_printer_friendly parameter of your plug-in function

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the render function or Ajax callback function of the plug-in.

```
apex_plugin_util.debug_region (
    p_plugin              => p_plugin,
    p_region              => p_region,
    p_is_printer_friendly => p_is_printer_friendly);
```

21.8 ESCAPE Function

This function is used if you have checked the standard attribute "Has Escape Output Attribute" option for your item type plug-in which allows a developer to decide if the output should be escaped or not.

Syntax

```
APEX_PLUGIN_UTIL.ESCAPE (
    p_value  IN VARCHAR2,
    p_escape IN BOOLEAN)
RETURN VARCHAR2;
```

Parameters

Table 21-7 ESCAPE Parameters

Parameter	Description
p_value	This is the value you want to escape depending on the p_escape parameter.
p_escape	If set to TRUE, the return value is escaped. If set to FALSE, the value is not escaped.

Example

This example outputs all values of the array l_display_value_list as a HTML list and escapes the value of the array depending on the setting the developer as picked when using the plug-in.

```
for i in 1 .. l_display_value_list.count
loop
    sys.htp.prn (
        '<li>'||
```

```
        apex_plugin_util.escape (
            p_value  => l_display_value_list(i),
            p_escape => p_item.escape_output )||
        '</li>' );
end loop;
```

21.9 EXECUTE_PLSQL_CODE Procedure

This procedure executes a PL/SQL code block and performs binding of bind variables in the provided PL/SQL code. This procedure is usually used for plug-in attributes of type PL/SQL Code.

Syntax

```
APEX_PLUGIN_UTIL.EXECUTE_PLSQL_CODE (
    p_plsql_code  IN VARCHAR2);
```

Parameters

Table 21-8 EXECUTE_PLSQL_CODE Parameters

Parameter	Description
p_plsql_code	PL/SQL code to be executed.

Example

Text which should be escaped and then printed to the HTTP buffer.

```
declare
    l_plsql_code VARCHAR(32767) := p_process.attribute_01;
begin
    apex_plugin_util.execute_plsql_code (
        p_plsql_code => l_plsql_code );
end;
```

21.10 GET_ATTRIBUTE_AS_NUMBER Function

This function returns the value of a plug-in attribute as a number, taking into account NLS decimal separator effective for the current database session. Use this function in plug-in PL/SQL source for custom attributes of type NUMBER instead of the built-in to_number function.

Syntax

```
APEX_PLUGIN_UTIL.GET_ATTRIBUTE_AS_NUMBER (
    p_value IN VARCHAR2 ),
    p_attribute_label IN VARCHAR2 )
    return NUMBER;
```

Parameters

Table 21-9 GET_ATTRIBUTE_AS_NUMBER Function Parameters

Parameter	Description
p_attribute_label	The label of the custom plug-in attribute.
p_value	The value of a custom attribute of type NUMBER.

Example

```
declare
    l_value number;
begin
    -- The following may fail for languages that don't use dot as the NLS decimal
separator
    l_value := to_number( p_region.attribute_04 );

    -- The following will work correctly regardless of the effective NLS decimal
separator
    l_value := apex_plugin_util.get_attribute_as_number( p_region.attribute_04,
'Minimum Amount' );
end;
/
```

21.11 GET_DATA Function Signature 1

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned as a string, independent of their data types. The search column is identified by providing a column number in the p_search_column_no parameter. This function takes into account character value comparison globalization attributes defined for the application.

Syntax

```
APEX_PLUGIN_UTIL.GET_DATA (
    p_sql_statement    IN VARCHAR2,
    p_min_columns      IN NUMBER,
    p_max_columns      IN NUMBER,
    p_component_name   IN VARCHAR2,
    p_search_type      IN VARCHAR2 DEFAULT 2,
    p_search_column_no IN VARCHAR2 DEFAULT 2,
    p_search_string    IN VARCHAR2 DEFAULT NULL,
    p_first_row        IN NUMBER DEFAULT NULL,
    p_max_rows         IN NUMBER DEFAULT NULL)
RETURN t_column_value_list;
```

Parameters

Table 21-10 GET_DATA Function Signature 1Parameters

Parameters	Description
p_sql_statement	SQL statement used for the lookup.

Table 21-10 (Cont.) GET_DATA Function Signature 1Parameters

Parameters	Description
p_min_columns	Minimum number of return columns.
p_max_columns	Maximum number of return columns.
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_search_type	Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns though p_max_columns range.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.

Return

Table 21-11 GET_DATA Function Signature 1 Return

Return	Description
t_column_value_list	Table of apex_application_global.vc_arr2 indexed by column number.

Example

The following example shows a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list.

```
function render_list (
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_column_value_list   apex_plugin_util.t_column_value_list;
begin
    l_column_value_list :=
        apex_plugin_util.get_data (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
```

```
            p_component_name   => p_item.name,
            p_search_type      => apex_plugin_util.c_search_contains_case,
            p_search_column_no => 1,
            p_search_string    => p_value );

    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list(1).count
    loop
        sys.htp.p(
            '<li>'||
            sys.htf.escape_sc(l_column_value_list(1)(i))|| -- display column
            '-'||
            sys.htf.escape_sc(l_column_value_list(2)(i))|| -- return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;
```

21.12 GET_DATA Function Signature 2

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned as a string, independent of their data types. The search column is identified by providing a column name in the `p_search_column_name` parameter. This function takes into account character value comparison globalization attributes defined for the application.

Syntax

```
APEX_PLUGIN_UTIL.GET_DATA (
    p_sql_statement      IN VARCHAR2,
    p_min_columns        IN NUMBER,
    p_max_columns        IN NUMBER,
    p_component_name     IN VARCHAR2,
    p_search_type        IN VARCHAR2 DEFAULT NULL,
    p_search_column_name IN VARCHAR2 DEFAULT NULL,
    p_search_string      IN VARCHAR2 DEFAULT NULL,
    p_first_row          IN NUMBER DEFAULT NULL,
    p_max_rows           IN NUMBER DEFAULT NULL)
RETURN t_column_value_list;
```

Parameters

Table 21-12 GET_DATA Function Signature 2 Parameters

Parameters	Description
`p_sql_statement`	SQL statement used for the lookup.
`p_min_columns`	Minimum number of return columns.
`p_max_columns`	Maximum number of return columns.
`p_component_name`	In case an error is returned, this is the name of the page item or report column used to display the error message.

Table 21-12 (Cont.) GET_DATA Function Signature 2 Parameters

Parameters	Description
p_search_type	Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore
p_search_column_name	This is the column name used to restrict the SQL statement.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.

Return

Table 21-13 GET_TABLE Function Signature 2

Parameter	Description
t_column_value_list	Table of apex_application_global.vc_arr2 indexed by column number.

Example

The following example shows a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list.

```
function render_list (
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_column_value_list   apex_plugin_util.t_column_value_list;
begin
    l_column_value_list :=
        apex_plugin_util.get_data (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
            p_component_name   => p_item.name,
            p_search_type      => apex_plugin_util.c_search_contains_case,
            p_search_column_name => 'ENAME',
            p_search_string    => p_value );

    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list(1).count
    loop
        sys.htp.p(
            '<li>'||
```

```
            sys.htf.escape_sc(l_column_value_list(1)(i))|| -- display column
            '-'||
            sys.htf.escape_sc(l_column_value_list(2)(i))|| -- return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;
```

21.13 GET_DATA2 Function Signature 1

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned along with their original data types. The search column is identified by providing a column number in the `p_search_column_no` parameter. This function takes into account character value comparison globalization attributes defines for the application.

Syntax

```
APEX_PLUGIN_UTIL.GET_DATA2 (
    p_sql_statement    IN VARCHAR2,
    p_min_columns      IN NUMBER,
    p_max_columns      IN NUMBER,
    p_data_type_list   IN WWV_GLOBAL.VC_ARR2 DEFAULT C_EMPTY_DATA_TYPE_LIST,
    p_component_name   IN VARCHAR2,
    p_search_type      IN VARCHAR2 DEFAULT 2,
    p_search_column_no IN VARCHAR2 DEFAULT 2,
    p_search_string    IN VARCHAR2 DEFAULT NULL,
    p_first_row        IN NUMBER DEFAULT NULL,
    p_max_rows         IN NUMBER DEFAULT NULL)
RETURN t_column_value_list2;
```

Parameters

Table 21-14 GET_DATA2 Parameters

Parameter	Description
`p_sql_statement`	SQL statement used for the lookup.
`p_min_columns`	Minimum number of return columns.
`p_max_columns`	Maximum number of return columns.
`p_data_type_list`	If provided, checks to make sure the data type for each column matches the specified data type in the array. Use the constants `c_data_type_*` for available data types.
`p_component_name`	In case an error is returned, this is the name of the page item or report column used to display the error message.
`p_search_type`	Must be one of the c_search_* constants. They are as follows: `c_search_contains_case`, `c_search_contains_ignore`, `c_search_exact_case`, `c_search_exact_ignore`

Table 21-14 (Cont.) GET_DATA2 Parameters

Parameter	Description
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns though p_max_columns range.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.

Return

Table 21-15 GET_DATA2 Return

Return	Description
t_column_value_list2	Table of t_column_values indexed by column number.

Example

The following example is a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list. This time, the first column of the LOV SQL statement is checked if it is of type VARCHAR2 and the second is of type number.

```
function render_list (
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_data_type_list    apex_application_global.vc_arr2;
    l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    -- The first LOV column has to be a string and the second a number
    l_data_type_list(1) := apex_plugin_util.c_data_type_varchar2;
    l_data_type_list(2) := apex_plugin_util.c_data_type_number;
    --
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
            p_data_type_list   => l_data_type_list,
            p_component_name   => p_item.name,
            p_search_type      => apex_plugin_util.c_search_contains_case,
            p_search_column_no => 1,
            p_search_string    => p_value );
    --
    sys.htp.p('<ul>');
```

```
    for i in 1 .. l_column_value_list.count(1)
    loop
        sys.htp.p(
            '<li>'||
            sys.htf.escape_sc(l_column_value_list(1).value_list(i).varchar2_value)||
-- display column
            '-'||
            sys.htf.escape_sc(l_column_value_list(2).value_list(i).number_value)||
-- return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;
```

The following example is a simple region type plug-in rendering function which executes the SQL query defined for the region. The result is then generated as a HTML list. This example demonstrates the advanced handling of object type columns like SDO_GEOMETRY.

```
function render (
    p_region in apex_plugin.t_region,
    p_plugin in apex_plugin.t_plugin,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_region_render_result
is
    l_column_value_list apex_plugin_util.t_column_value_list2;
    l_geometry sdo_geometry;
    l_value varchar2(32767);
    l_dummy pls_integer;
begin
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement => p_region.source,
            p_min_columns => 1,
            p_max_columns => null,
            p_component_name => p_region.name );
   --
   sys.htp.p('<ul>');
   for row in 1 .. l_column_value_list(1).value_list.count loop

       sys.htp.p('<li>');

       for col in 1 .. l_column_value_list.count loop
            if l_column_value_list(col).data_type = 'SDO_GEOMETRY' then

               -- Object Type columns are always returned using ANYDATA and we have
to
               -- use GETOBJECT to transform them back into the original object type
               l_dummy :=
l_column_value_list(col).value_list(row).anydata_value.getobject(
l_geometry );
               l_value := '( type=' || l_geometry.sdo_gtype || ' srid=' ||
l_geometry.sdo_srid ||
                          case when l_geometry.sdo_point is not null then
                               ',x=' || l_geometry.sdo_point.x ||
                               ',y=' || l_geometry.sdo_point.y ||
                               ',z=' || l_geometry.sdo_point.z
                          end ||
                          ' )';
               else
                          l_value := apex_plugin_util.get_value_as_varchar2(
```

```
                                    p_data_type => l_column_value_list(col).data_type,
                                    p_value =>
l_column_value_list(col).value_list(row) );
               end if;

               sys.htp.p( case when col > 1 then ' - ' end || l_value );
          end loop;

          sys.htp.p('<li>');
      end loop;
      sys.htp.p('<ul>');

      return null;
end;
```

21.14 GET_DATA2 Function Signature 2

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned along with their original data types. The search column is identified by providing a column number in the `p_search_column_no` parameter. This function takes into account character value comparison globalization attributes defines for the application.

Syntax

```
APEX_PLUGIN_UTIL.GET_DATA2 (
    p_sql_statement      IN VARCHAR2,
    p_min_columns        IN NUMBER,
    p_max_columns        IN NUMBER,
    p_data_type_list     IN WWV_GLOBAL.VC_ARR2 DEFAULT C_EMPTY_DATA_TYPE_LIST,
    p_component_name     IN VARCHAR2,
    p_search_type        IN VARCHAR2 DEFAULT 2,
    p_search_column_name IN VARCHAR2 DEFAULT 2,
    p_search_string      IN VARCHAR2 DEFAULT NULL,
    p_first_row          IN NUMBER DEFAULT NULL,
    p_max_rows           IN NUMBER DEFAULT NULL)
RETURN t_column_value_list2;
```

Parameters

Table 21-16 GET_DATA2 Function Signature 2

Parameter	Description
`p_sql_statement`	SQL statement used for the lookup.
`p_min_columns`	Minimum number of return columns.
`p_max_columns`	Maximum number of return columns.
`p_data_type_list`	If provided, checks to make sure the data type for each column matches the specified data type in the array. Use the constants `c_data_type_*` for available data types.
`p_component_name`	In case an error is returned, this is the name of the page item or report column used to display the error message.

Table 21-16 (Cont.) GET_DATA2 Function Signature 2

Parameter	Description
p_search_type	Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore
p_search_column_name	The column name used to restrict the SQL statement.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.

Return

Table 21-17 GET_DATA2 Function Signature 2 Return

Parameter	Description
t_column_value_list2	Table of t_column_values indexed by column number.

Example

The following example is a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list. This time, the first column of the LOV SQL statement is checked if it is of type VARCHAR2 and the second is of type number.

```
function render_list (
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_data_type_list    apex_application_global.vc_arr2;
    l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    -- The first LOV column has to be a string and the second a number
    l_data_type_list(1) := apex_plugin_util.c_data_type_varchar2;
    l_data_type_list(2) := apex_plugin_util.c_data_type_number;
    --
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
            p_data_type_list   => l_data_type_list,
            p_component_name   => p_item.name,
            p_search_type      => apex_plugin_util.c_search_contains_case,
            p_search_column_name => 'ENAME',
```

```
            p_search_string    => p_value );
    --
    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list.count(1)
    loop
        sys.htp.p(
            '<li>'||
            sys.htf.escape_sc(l_column_value_list(1).value_list(i).varchar2_value)||
-- display column
            '-'||
            sys.htf.escape_sc(l_column_value_list(2).value_list(i).number_value)||
-- return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;
```

21.15 GET_DISPLAY_DATA Function Signature 1

This function gets the display lookup value for the value specified in `p_search_string`.

Syntax

```
APEX_PLUGIN_UTIL.GET_DISPLAY_DATA (
    p_sql_statement    IN VARCHAR2,
    p_min_columns      IN NUMBER,
    p_max_columns      IN NUMBER,
    p_component_name   IN VARCHAR2,
    p_display_column_no IN BINARY_INTEGER DEFAULT 1,
    p_search_column_no  IN BINARY_INTEGER DEFAULT 2,
    p_search_string    IN VARCHAR2 DEFAULT NULL,
    p_display_extra    IN BOOLEAN DEFAULT TRUE)
RETURN VARCHAR2;
```

Parameters

Table 21-18 GET_DISPLAY_DATA Signature 1 Parameters

Parameter	Description
`p_sql_statement`	SQL statement used for the lookup.
`p_min_columns`	Minimum number of return columns.
`p_max_columns`	Maximum number of return columns.
`p_component_name`	In case an error is returned, this is the name of the page item or report column used to display the error message.
`p_display_column_no`	Number of the column returned from the SQL statement. Must be within the `p_min_columns` though `p_max_columns` range
`p_search_column_no`	Number of the column used to restrict the SQL statement. Must be within the `p_min_columns` though `p_max_columns` range.
`p_search_string`	Value used to restrict the query.

Table 21-18 (Cont.) GET_DISPLAY_DATA Signature 1 Parameters

Parameter	Description
p_display_extra	If set to TRUE, and a value is not found, the search value is added to the result instead.

Return

Table 21-19 GET_DISPLAY_DATA Signature 1 Return

Return	Description
VARCHAR2	Value of the first record of the column specified by p_display_column_no. If no record was found it contains the value of p_search_string if the parameter p_display_extra is set to TRUE. Otherwise NULL is returned.

Example

The following example does a lookup with the value provided in p_value and returns the display column of the LOV query.

```
function render_value (
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
begin
    sys.htp.p(sys.htf.escape_sc(
        apex_plugin_util.get_display_data (
            p_sql_statement     => p_item.lov_definition,
            p_min_columns       => 2,
            p_max_columns       => 2,
            p_component_name    => p_item.name,
            p_display_column_no => 1,
            p_search_column_no  => 2,
            p_search_string     => p_value )));
end render_value;
```

21.16 GET_DISPLAY_DATA Function Signature 2

This function looks up all the values provided in the p_search_value_list instead of just a single value lookup.

Syntax

```
APEX_PLUGIN_UTIL.GET_DISPLAY_DATA (
    p_sql_statement    IN VARCHAR2,
    p_min_columns      IN NUMBER,
    p_max_columns      IN NUMBER,
    p_component_name   IN VARCHAR2,
    p_display_column_no IN BINARY_INTEGER DEFAULT 1,
```

```
    p_search_column_no  IN BINARY_INTEGER DEFAULT 2,
    p_search_value_list IN ww_flow_global.vc_arr2,
    p_display_extra     IN BOOLEAN DEFAULT TRUE)
RETURN apex_application_global.vc_arr2;
```

Parameters

Table 21-20 GET_DISPLAY_DATA Signature 2 Parameters

Parameter	Description
p_sql_statement	SQL statement used for the lookup.
p_min_columns	Minimum number of return columns.
p_max_columns	Maximum number of return columns.
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_display_column_no	Number of the column returned from the SQL statement. Must be within the p_min_columns though p_max_columns range.
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns though p_max_columns range.
p_search_value_list	Array of values to look up.
p_display_extra	If set to TRUE, and a value is not found, the search value is added to the result instead.

Return

Table 21-21 GET_DISPLAY_DATA Signature 2 Return

Return	Description
apex_application_global .vc_arr2	List of VARCHAR2 indexed by pls_integer. For each entry in p_search_value_list the resulting array contains the value of the first record of the column specified by p_display_column_no in the same order as in p_search_value_list. If no record is found it contains the value of p_search_string if the parameter p_display_extra is set to TRUE. Otherwise the value is skipped.

Example

Looks up the values 7863, 7911 and 7988 and generates a HTML list with the value of the corresponding display column in the LOV query.

```
function render_list (
    p_plugin              in apex_plugin.t_plugin,
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
```

```
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_search_list apex_application_global.vc_arr2;
    l_result_list apex_application_global.vc_arr2;
begin
    l_search_list(1) := '7863';
    l_search_list(2) := '7911';
    l_search_list(3) := '7988';
    --
    l_result_list :=
        apex_plugin_util.get_display_data (
            p_sql_statement     => p_item.lov_definition,
            p_min_columns       => 2,
            p_max_columns       => 2,
            p_component_name    => p_item.name,
            p_search_column_no  => 1,
            p_search_value_list => l_search_list );
    --
    sys.htp.p('<ul>');
    for i in 1 .. l_result_list.count
    loop
        sys.htp.p(
            '<li>'||
            sys.htf.escape_sc(l_result_list(i))||
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;
```

21.17 GET_ELEMENT_ATTRIBUTES Function

This function returns some of the standard attributes of an HTML element (for example, id, name, required, placeholder, aria-error-attributes, class) which is used if a HTML input/select/textarea/... tag is generated to get a consistent set of attributes.

Syntax

```
APEX_PLUGIN_UTIL.GET_ELEMENT_ATTRIBUTES (
    p_item in apex_plugin.t_page_item,
    p_name in varchar2 default null,
    p_default_class in varchar2 default null,
    p_add_id in boolean default true,
    p_add_labelledby in boolean default true )
    return varchar2;
```

Parameters

Table 21-22 GET_ELEMENT_ATTRIBUTES Function Parameters

Parameters	Description
p_add_labelled_by	Returns some of the general attributes of an HTML element (for example, the ID, name, required, placeholder, aria-error-attributes, class) which should be used if an HTML input, select, or textarea tag is generated to get a consistent set of attributes.Set to FALSE if you render a HTML input element like input, select, or textarea which does not require specifying the aria-labelledby attribute because the label's `for` attribute works for those HTML input elements. Set it to TRUE for all 'non-standard form element widgets (that is, those using div, span, and so on.) which do allow focus to make them accessible to screen readers. **Note:** Inclusion of aria-labelled by is also dependent on the item plug-in having Standard Form Element set to No and that there is a `#LABEL_ID#` substitution defined in the item's corresponding label template.
p_item	This is the p_item parameter of your plug-in function.
p_name	This is the value which has been return by apex_plugin.get_input_name_or_page_item
p_default_class	Default CSS class which which should be contained in the result string.
p_add_id	If set to TRUE then the id attribute is also contained in the result string.

Example

This example emits an INPUT tag of type text which uses `apex_plugin_util.get_element_attributes` to automatically include the most common attributes.

```
sys.htp.prn (
        '<input type="text" ' ||
        apex_plugin_util.get_element_attributes(p_item, l_name, 'text_field') ||
        'value="'||l_escaped_value||'" '||
        'size="'||p_item.element_width||'" '||
        'maxlength="'||p_item.element_max_length||'" '||
        ' />');
```

21.18 GET_PLSQL_EXPRESSION_RESULT Function

This function executes a PL/SQL expression and returns a result. This function also performs the binding of any bind variables in the provided PL/SQL expression. This function is usually used for plug-in attributes of type PL/SQL Expression.

Syntax

```
APEX_PLUGIN_UTIL.GET_PLSQL_EXPRESSION_RESULT (
    p_plsql_expression IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 21-23 GET_PLSQL_EXPRESSION_RESULT Parameters

Parameter	Description
p_plsql_expression_result	A PL/SQL expression that returns a string.

Return

Table 21-24 GET_PLSQL_EXPRESSION_RESULT Return

Return	Description
VARCHAR2	String result value returned by the PL/SQL Expression.

Example

This example executes and returns the result of the PL/SQL expression which is specified in `attribute_03` of an item type plug-in attribute of type "PL/SQL Expression".

```
l_result := apex_plugin_util.get_plsql_expression_result (
    p_plsql_expression => p_item.attribute_03 );
```

21.19 GET_PLSQL_FUNCTION_RESULT Function

This function executes a PL/SQL function block and returns the result. This function also performs binding of bind variables in the provided PL/SQL Function Body. This function is usually used for plug-in attributes of type PL/SQL Function Body.

Syntax

```
APEX_PLUGIN_UTIL.GET_PLSQL_FUNCTION_RESULT (
    p_plsql_function IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 21-25 GET_PLSQL_FUNCTION_RESULT Parameters

Parameter	Description
p_plsql_function	A PL/SQL function block that returns a result of type string.

Return

Table 21-26 GET_PLSQL_FUNCTION_RESULT Return

Return	Description
VARCHAR2	String result value returned by the PL/SQL function block.

Example

The following example executes and returns the result of the PL/SQL function body that is specified in `attribute_03` of an item type plug-in attribute of type PL/SQL Function Body.

```
l_result := apex_plugin_util.get_plsql_function_result (
    p_plsql_function => p_item.attribute_03 );
```

21.20 GET_POSITION_IN_LIST Function

This function returns the position in the list where p_value is stored. If it is not found, null is returned.

Syntax

```
APEX_PLUGIN_UTIL.GET_POSITION_IN_LIST(
    p_list IN apex_application_global.vc_arr2,
    p_value IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 21-27 GET_POSITION_IN_LIST Parameters

Parameter	Description
`p_list`	Array of type `apex_application_global.vc_arr2` that contains entries of type VARCHAR2.
`p_value`	Value located in the `p_list` array.

Return

Table 21-28 GET_POSITION_IN_LIST Return

Return	Description
`NUMBER`	Returns the position of `p_value` in the array `p_list`. If it is not found `NULL` is returned.

Example

The following example searchs for "New York" in the provided list and returns 2 into `l_position`.

```
declare
    l_list     apex_application_global.vc_arr2;
```

```
    l_position number;
begin
    l_list(1) := 'Rome';
    l_list(2) := 'New York';
    l_list(3) := 'Vienna';

    l_position := apex_plugin_util.get_position_in_list (
                      p_list  => l_list,
                      p_value => 'New York' );
end;
```

21.21 GET_SEARCH_STRING Function

Based on the provided value in `p_search_type` the passed in value of `p_search_string` is returned unchanged or is converted to uppercase. Use this function with the `p_search_string` parameter of `get_data` and `get_data2`.

Syntax

```
APEX_PLUGIN_UTIL.GET_SEARCH_STRING(
    p_search_type IN VARCHAR2,
    p_search_string IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 21-29 GET_SEARCH_STRING Parameters

Parameter	Description
`p_search_type`	Type of search when used with `get_data` and `get_data2`. Use one of the `c_search_*` constants.
`p_search_string`	Search string used for the search with `get_data` and `get_data2`.

Return

Table 21-30 GET_SEARCH_STRING Return

Return	Description
VARCHAR2	Returns `p_search_string` unchanged or in uppercase if `p_search_type` is of type `c_search_contains_ignore` or `c_search_exact_ignore`.

Example

This example uses a call to `get_data` or `get_data2` to make sure the search string is using the correct case.

```
l_column_value_list :=
    apex_plugin_util.get_data (
        p_sql_statement    => p_item.lov_definition,
        p_min_columns      => 2,
        p_max_columns      => 2,
        p_component_name   => p_item.name,
```

```
        p_search_type      => apex_plugin_util.c_search_contains_ignore,
        p_search_column_no => 1,
        p_search_string    => apex_plugin_util.get_search_string (
            p_search_type   => apex_plugin_util.c_search_contains_ignore,
            p_search_string => p_value ) );
```

21.22 GET_VALUE_AS_VARCHAR2 Function

This function can be used if you use GET_DATA2 to read the column values along with their original data types. It will convert and return the passed in p_value as VARCHAR2.

Syntax

```
function get_value_as_varchar2 (
   p_data_type in varchar2,
   p_value in t_value,
   p_format_mask in varchar2 default null )
   return varchar2;
```

Parameters

Table 21-31 GET_VALUE_AS_VARCHAR2 Function Parameters

Parameter	Description
p_data_type	The data type of the value stored in p_value.
p_value	The value of type t_value which contains the value to be converted and returned as VARCHAR2.
p_format_mask	The format mask used to convert the value into a VARCHAR2.

Example

The following example emits all values stored in the data type aware l_column_value_list array as VARCHAR2.

```
declare
  l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    -- Populate l_column_value_list by calling apex_plugin_util.get_data2
    ...
    -- Emit returned data
    sys.htp.p( '<table>' );
    for l_row in 1 .. l_column_value_list( 1 ).value_list.count
    loop
        sys.htp.p( '<tr>' );
        for l_column in 1 .. l_column_value_list.count loop
            sys.htp.p (
                '<td>' ||
                apex_plugin_util.get_value_as_varchar2 (
                    p_data_type => l_column_value_list( l_column ).data_type,
                    p_value => l_column_value_list( l_column ).value_list( l_row )
                    ) ||
```

```
                '</td>' );
        end loop;
        sys.htp.p( '</tr>' );
    end loop;
    sys.htp.p( '</table>' );
end;
```

21.23 IS_EQUAL Function

This function returns TRUE if both values are equal and FALSE if not. If both values are NULL, TRUE is returned.

Syntax

```
APEX_PLUGIN_UTIL.IS_EQUAL (
    p_value1 IN VARCHAR2
    p_value2 IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 21-32 IS_EQUAL Parameters

Parameter	Description
p_value1	First value to compare.
p_value2	Second value to compare.

Return

Table 21-33 IS_EQUAL Return

Return	Description
BOOLEAN	Returns TRUE if both values are equal or both values are NULL, otherwise it returns FALSE.

Example

In the following example, if the value in the database is different from what is entered, the code in the if statement is executed.

```
if NOT apex_plugin_util.is_equal(l_database_value, l_current_value) then
    -- value has changed, do something
    null;
end if;
```

21.24 PAGE_ITEM_NAMES_TO_JQUERY Function

This function returns a jQuery selector based on a comma delimited string of page item names. For example, you could use this function for a plug-in attribute called "Page Items to Submit" where the JavaScript code has to read the values of the specified page items.

Syntax

```
APEX_PLUGIN_UTIL.PAGE_ITEM_NAMES_TO_JQUERY (
    p_page_item_names IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 21-34 PAGE_ITEM_NAMES_TO_JQUERY Parameters

Parameter	Description
p_page_item_names	Comma delimited list of page item names.

Return

Table 21-35 PAGE_ITEM_NAMES_TO_JQUERY Return

Return	Description
VARCHAR2	Transforms the page items specified in p_page_item_names into a jQuery selector.

Example

The following example shows the code to construct the initialization call for a JavaScript function called myOwnWidget. This function gets an object with several attributes where one attribute is pageItemsToSubmit which is expected to be a jQuery selector.

```
apex_javascript.add_onload_code (
    p_code => 'myOwnWidget('||
                  '"#'||p_item.name||'",'||
                  '{'||
                  apex_javascript.add_attribute('ajaxIdentifier',
apex_plugin.get_ajax_identifier)||
                  apex_javascript.add_attribute('dependingOnSelector',
apex_plugin_util.page_item_names_to_jquery(p_item.lov_cascade_parent_items))||
                  apex_javascript.add_attribute('optimizeRefresh',
p_item.ajax_optimize_refresh)||
                  apex_javascript.add_attribute('pageItemsToSubmit',
apex_plugin_util.page_item_names_to_jquery(p_item.ajax_items_to_submit))||
                  apex_javascript.add_attribute('nullValue',
p_item.lov_null_value, false, false)||
                  '});' );
```

21.25 PRINT_DISPLAY_ONLY Procedure

This procedure outputs a SPAN tag for a display only field.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_DISPLAY_ONLY (
    p_item_name        IN VARCHAR2,
    p_display_value    IN VARCHAR2,
    p_show_line_breaks IN BOOLEAN,
    p_attributes       IN VARCHAR2,
    p_id_postfix       IN VARCHAR2 DEFAULT '_DISPLAY');
```

Parameters

Table 21-36 PRINT_DISPLAY_ONLY Parameter

Parameter	Description
p_item_name	Name of the page item. This parameter should be called with p_item.name.
p_display_value	Text to be displayed.
p_show_line_breaks	If set to TRUE line breaks in p_display_value are changed to so that the browser renders them as line breaks.
p_attributes	Additional attributes added to the SPAN tag.
p_id_postfix	Postfix which is getting added to the value in p_item_name to get the ID for the SPAN tag. Default is _DISPLAY.

Example

The following code could be used in an item type plug-in to render a display only page item.

```
apex_plugin_util.print_display_only (
    p_item_name        => p_item.name,
    p_display_value    => p_value,
    p_show_line_breaks => false,
    p_escape           => true,
    p_attributes       => p_item.element_attributes );
```

21.26 PRINT_ESCAPED_VALUE Procedure

This procedure outputs the value in an escaped form and chunks big strings into smaller outputs.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_ESCAPED_VALUE (
    p_value   IN VARCHAR2);
```

Parameters

Table 21-37 PRINT_ESCAPED_VALUE Parameter

Parameter	Description
p_value	Text which should be escaped and then printed to the HTTP buffer.

Example

Prints a hidden field with the current value of the page item.

```
sys.htp.prn('<input type="hidden" name="'||l_name||'" id="'||p_item_name||'"
value="');
print_escaped_value(p_value);
sys.htp.prn('">');
```

21.27 PRINT_HIDDEN_IF_READONLY Procedure

This procedure outputs a hidden field to store the page item value if the page item is rendered as readonly and is not printer friendly. If this procedure is called in an item type plug-in, the parameters of the plug-in interface should directly be passed in.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_HIDDEN_IF_READ_ONLY (
    p_item_name   IN VARCHAR2,
    p_value       IN VARCHAR2,
    p_is_readonly IN BOOLEAN,
    p_is_printer_friendly IN BOOLEAN,
    p_id_postfix  IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-38 PRINT_HIDDEN_IF_READONLY Parameters

Parameter	Description
p_item_name	Name of the page item. For this parameter the p_item.name should be passed in.
p_value	Current value of the page item. For this parameter p_value should be passed in.
p_is_readonly	Is the item rendered readonly. For this parameter p_is_readonly should be passed in.
p_is_printer_friendly	Is the item rendered in printer friendly mode. For this parameter p_is_printer_friendly should be passed in.
p_id_postfix	Used to generate the ID attribute of the hidden field. It is build based on p_item_name and the value in p_id_postfix.

Example

Writes a hidden field with the current value to the HTTP output if p_is_readonly is TRUE and p_printer_friendly is FALSE.

```
apex_plugin_util.print_hidden_if_readonly (
    p_item_name           => p_item.name,
    p_value               => p_value,
    p_is_readonly         => p_is_readonly,
    p_is_printer_friendly => p_is_printer_friendly );
```

21.28 PRINT_JSON_HTTP_HEADER Procedure

This procedure outputs a standard HTTP header for a JSON output.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_JSON_HTTP_HEADER;
```

Parameters

None.

Example

This example shows how to use this procedure in the Ajax callback function of a plugin. This code outputs a JSON structure in the following format:
`[{"d":"Display 1","r":"Return 1"},{"d":"Display 2","r":"Return 2"}]`

```
-- Write header for the JSON stream.
apex_plugin_util.print_json_http_header;
-- initialize the JSON structure
sys.htp.p('[');
-- loop through the value array
for i in 1 .. l_values.count
loop
    -- add array entry
    sys.htp.p (
        case when i > 1 then ',' end||
        '{'||
        apex_javascript.add_attribute('d',
sys.htf.escape_sc(l_values(i).display_value), false, true)||
        apex_javascript.add_attribute('r',
sys.htf.escape_sc(l_values(i).return_value), false, false)||
        '}' );
end loop;
-- close the JSON structure
sys.htp.p(']');
```

21.29 PRINT_LOV_AS_JSON Procedure

This procedure outputs a JSON response based on the result of a two column LOV in the format:

```
[{"d:"display","r":"return"},{"d":....,"r":....},....]
```

> **Note:**
>
> The HTTP header is initialized with MIME type "application/json" as well.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_LOV_AS_JSON (
    p_sql_statement         IN VARCHAR2,
    p_component_name        IN VARCHAR2,
    p_escape                IN BOOLEAN,
    p_replace_substitutions IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 21-39 PRINT_LOV_AS_JSON Parameters

Parameter	Description
p_sql_statement	A SQL statement which returns two columns from the SELECT.
p_component_name	The name of the page item or report column that is used in case an error is displayed.
p_escape	If set to TRUE the value of the display column is escaped, otherwise it is output as is.
p_replace_substitutions	If set to TRUE, apex_plugin_util.replace_substitutions is called for the value of the display column, otherwise, it is output as is.

Example

This example shows how to use the procedure in an Ajax callback function of an item type plug-in. The following call writes the LOV result as a JSON array to the HTTP output.

```
apex_plugin_util.print_lov_as_json (
    p_sql_statement  => p_item.lov_definition,
    p_component_name => p_item.name,
    p_escape         => true );
```

21.30 PRINT_OPTION Procedure

This procedure outputs an OPTION tag.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_OPTION (
    p_display_value       IN VARCHAR2,
    p_return_value        IN VARCHAR2,
    p_is_selected         IN BOOLEAN,
    p_attributes          IN VARCHAR2,
    p_escape              IN BOOLEAN DEFAULT TRUE);
```

Parameters

Table 21-40 PRINT_OPTION Parameters

Parameter	Description
p_display_value	Text which is displayed by the option.
p_return_value	Value which is set when the option is picked.
p_is_selected	Set to TRUE if the selected attribute should be set for this option.

Table 21-40 (Cont.) PRINT_OPTION Parameters

Parameter	Description
p_attributes	Additional HTML attributes which should be set for the OPTION tag.
p_escape	Set to TRUE if special characters in p_display_value should be escaped.

Example

The following example could be used in an item type plug-in to create a SELECT list. Use apex_plugin_util.is_equal to find out which list entry should be marked as current.

```
sys.htp.p('<select id="'||p_item.name||'" size="'||nvl(p_item.element_height, 5)||'"
'||coalesce(p_item.element_attributes, 'class="new_select_list"')||'>');
-- loop through the result and add list entries
for i in 1 .. l_values.count
loop
    apex_plugin_util.print_option (
        p_display_value => l_values(i).display_value,
        p_return_value  => l_values(i).return_value,
        p_is_selected   => apex_plugin_util.is_equal(l_values(i).return_value,
p_value),
        p_attributes    => p_item.element_option_attributes,
        p_escape        => true );
end loop;
sys.htp.p('</select>');
```

21.31 REPLACE_SUBSTITUTIONS Function

This function replaces any &ITEM. substitution references with their actual value. If p_escape is set to TRUE, any special characters contained in the value of the referenced item are escaped to prevent Cross-site scripting (XSS) attacks.

Syntax

```
apex_plugin_util.replace_substitutions (
    p_value    in varchar2,
    p_escape   in boolean default true )
    return varchar2;
```

Parameters

Table 21-41 REPLACE_SUBSTITUTION Parameters

Parameter	Description
p_value	This value is a string which can contain several &ITEM. references which are replaced by their actual page item values.

Table 21-41 (Cont.) REPLACE_SUBSTITUTION Parameters

Parameter	Description
p_escape	If set to TRUE any special characters contained in the value of the referenced item are escaped to prevent Cross-site scripting (XSS) attacks. If set to FALSE, the referenced items are not escaped.

Example

The following example replaces any substitution syntax references in the region plug-in `attribute 05` with their actual values. Any special characters in the values are escaped.

```
l_advanced_formatting  := apex_plugin_util.replace_substitutions (
                              p_value => p_region.attribute_05,
                              p_escape => true );
```

21.32 SET_COMPONENT_VALUES Procedure

This procedure extends `Session State` to include the column values of a specific row number. By doing so, columns can be referenced using `substitution syntax` or the V function in the same way as you can reference page or application items.

Note:

Always call `apex_plugin_util.clear_component_values` after you are done processing the current row!

Syntax

```
procedure set_component_values (
    p_column_value_list in t_column_list,
    p_row_num           in pls_integer );
```

Parameters

Table 21-42 SET_COMPONENT_VALUES Parameters

Parameter	Description
p_column_value_list	Table of `t_column_values returned` by the call to `apex_plugin_util.get_data2`.
p_row_num	Row number in `p_column_value_list` for which the column values should be set in `Session State`.

Example

This example is the selection of a simple item type plug-in rendering function which renders a link list based on a provided SQL query. Instead of a fixed SQL query format where the first column contains the link and the second contains the link label, it

allows a developer using this plug-in to enter any SQL statement and then use substitution syntax to reference the values of the executed SQL query.

```
function render_link_list (
    p_item                in apex_plugin.t_page_item,
    p_value               in varchar2,
    p_is_readonly         in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
  -- The link target plug-in attribute 01 would allow that a developer can enter a
link which references columns
  -- of the provided SQL query using subsitution syntax.
  -- For example: f?p=&APP_ID.:1:&APP_SESSION.::&DEBUG.::P1_EMPNO:&EMPNO.
  -- where &EMPNO. references the column EMPNO in the SQL query.
  c_link_target    constant varchar2(4000) := p_item.attribute_01;
  -- The link label column plug-in attribute 02 would allows a developer to
reference a column of the SQL query
  -- which should be used as the text for the link.
  c_link_label_column constant varchar2(128) := p_item.attribute_02;
  --
  l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement =>
            ... );
    --
    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list.count(1)
    loop
        -- Set all column values of the current row
        apex_plugin_util.set_component_values (
            p_column_value_list => l_column_value_list,
            p_row_num           => i );
        --
        sys.htp.p(
            '<li><a href="' ||
            apex_escape.html_attribute( apex_util.prepare_url( c_link_target )) ||
'">' ||
            apex_escape.html( v( c_link_label_column )) ||
       '</a></li>');

        --
        apex_plugin_util.clear_component_values;
    end loop;
    sys.htp.p('<ul>');
end;
```

22

APEX_REGION

The APEX_REGION package is the public API for handling regions.

22.1 IS_READ_ONLY Function

This function returns TRUE if the current region is rendered read-only and FALSE if region is not rendered read-only. If the function is called from a context where no region is currently processed, it returns NULL. For example, you can use this function in conditions of a region or its underlying items and buttons.

Syntax

```
FUNCTION IS_READ_ONLY
RETURN BOOLEAN;
```

Parameters

None.

Example

This examples purges the session for a specific region cache for the whole application.

```
RETURN APEX_REGION.IS_READ_ONLY;
```

22.2 PURGE_CACHE Procedure

This procedure purges the region cache of the specified application, page, and region.

Syntax

```
PROCEDURE PURGE_CACHE (
    p_application_id       IN NUMBER DEFAULT apex.g_flow_id,
    p_page_id              IN NUMBER DEFAULT NULL,
    p_region_id            IN NUMBER DEFAULT NULL,
    p_current_session_only IN BOOLEAN DEFAULT FALSE );
```

Parameters

Table 22-1 PURGE_CACHE Parameters

Parameter	Description
`p_application_id`	`ID` of the application where the region caches should be purged. Defaults to the current application.
`p_page_id`	`ID` of the page where the region caches should be purged. If no value is specified (which is the default), all regions of the application are purged.
`p_region_id`	ID of a specific region on a page. If no value is specified, all regions of the specified page are purged.
`p_current_session_only`	Specify true if you only want to purge entries that where saved for the current session. Defaults to false.

Example

This example purges session specific region cache for the whole application.

```
BEGIN
     APEX_REGION.PURGE_CACHE (
         p_current_session_only => true );
END;
```

23

APEX_SESSION

The package enables you to configure Application Express sessions.

23.1 SET_DEBUG Procedure

This procedure sets debug level for all future requests in a session.

Syntax

```
procedure set_debug (
   p_session_id in number default apex.g_instance,
   p_level in apex_debug_api.t_log_level );
```

Parameters

Table 23-1 SET_DEBUG Procedure Parameters

Parameters	Description
p_session_id	The session id. **Note:** The session must belong to the current workspace or the caller must be able to set the session's workspace.
p_level	The debug level. 0 or NULL disables debug, 1-9 sets a debug level.

Example 1

This example shows how to set debug for session 1234 to INFO level.

```
apex_session.set_debug (
    p_session_id => 1234,
         p_level => apex_debug.c_log_level_info );
commit;
```

Example 2

This example shows how to disable debug in session 1234.

```
apex_session.set_debug (
    p_session_id => 1234,
```

```
        p_level => null );
commit;
```

See Also:

- "ENABLE Procedure (page 8-3)"
- "DISABLE Procedure (page 8-2)"

23.2 SET_TRACE Procedure

This procedure sets trace mode in all future requests of a session.

Syntax

```
procedure set_trace (
   p_session_id in number default apex.g_instance,
   p_mode in varchar2 );
```

Parameters

Table 23-2 SET_TRACE Procedure Parameters

Parameters	Description
p_session_id	The session id. **Note:** The session must belong to the current workspace or the caller must be able to set the session's workspace.
p_level	The trace mode. NULL disables trace, SQL enables SQL trace.

Example 1

This example shows how to enable trace in requests for session 1234.

```
apex_session.set_trace (
    p_session_id => 1234,
          p_mode => 'SQL' );
commit;
```

Example 2

This example shows how to disable trace in requests for session 1234.

```
apex_session.set_trace (
    p_session_id => 1234,
          p_mode => null );
commit;
```

24

APEX_SPATIAL

This package enables you to use Oracle Locator and the Spatial Option within Application Express. In an Application Express context, the logon user of the database session is typically `APEX_PUBLIC_USER` or `ANONYMOUS`. Spatial developers can not directly use DML on `USER_SDO_GEOM_METADATA` within such a session, for example, in SQL Commands within SQL Workshop. The Spatial view's trigger performs DML as the logon user, but it has to run as the application owner or workspace user. With the `APEX_SPATIAL` API, developers can use the procedures and functions below to insert, update and delete rows of `USER_SDO_GEOM_METADATA` as the current Application Express user. The package also provides a few utilities that simplify the use of Spatial in Application Express.

24.1 Data Types

The data types used by this package are described in this section.

t_srid

```
subtype t_srid is number;
```

c_no_reference_system

```
c_no_reference_system constant t_srid := null;
```

c_wgs_84

```
c_wgs_84  constant t_srid := 4326; -- World Geodetic System, EPSG:4326
```

24.2 CHANGE_GEOM_METADATA Procedure

This procedure modifies a spatial metadata record.

Syntax

```
APEX_SPATIAL.CHANGE_GEOM_METADATA (
    p_table_name        IN VARCHAR2,
    p_column_name       IN VARCHAR2,
    p_new_table_name    IN VARCHAR2 DEFAULT NULL,
    p_new_column_name   IN VARCHAR2 DEFAULT NULL,
    p_diminfo           IN mdsys.sdo_dim_array,
    p_srid              IN t_srid );
```

Parameters

Table 24-1 CHANGE_GEOM_METADATA Parameters

Parameter	Description
`p_table_name`	Name of the feature table.
`p_column_name`	Name of the column of type `mdsys.sdo_geometry`.
`p_new_table_name`	New name of a feature table (or null, to keep the current value).
`p_new_column_name`	New name of the column of type `mdsys.sdo_geometry` (or null, to keep the current value.
`p_diminfo`	`SDO_DIM_ELEMENT` array, ordered by dimension, with one entry for each dimension.
`p_srid`	`SRID` value for the coordinate system for all geometries in the column.

Example

The code below modifies the dimensions of column `CITIES.SHAPE.`

```
begin
    for l_meta in ( select *
                      from user_sdo_geom_metadata
                     where table_name  = 'CITIES'
                       and column_name = 'SHAPE' )
    loop
        apex_spatial.change_geom_metadata (
            p_table_name  => l_meta.table_name,
            p_column_name => l_meta.column_name,
            p_diminfo     => SDO_DIM_ARRAY (
                               SDO_DIM_ELEMENT('X', -180, 180, 0.1),
                                      SDO_DIM_ELEMENT('Y',  -90,  90, 0.1) ),
        p_srid        => l_meta.srid );
    end loop;
end;
```

24.3 CIRCLE_POLYGON Function

This function creates a polygon that approximates a circle at (`p_lon`, `p_lat`) with radius of `p_radius`. See `mdsys.sdo_util.circle_polygon` for details.

Syntax

```
APEX_SPATIAL.CIRCLE_POLYGON (
    p_lon        IN NUMBER,
    p_lat        IN NUMBER,
    p_radius     IN NUMBER,
    p_arc_tolerance  IN NUMBER DEFAULT 20,
    p_srid        IN t_srid DEFAULT c_wgs_84 )
RETURN mdsys.sdo_geometry;
```

Parameters

Table 24-2 CIRCLE_POLYGON Parameters

Parameter	Description
p_lon	Longitude position of the lower left point.
p_lat	Latitude position of the lower left point.
`p_radius`	Radius of the circle in meters.
`p_arc_tolerance`	Arc tolerance (default 20).
p_srid	Reference system (default c_wgs_84).

Returns

Table 24-3 CIRCLE_POLYGON Function Returns

Return	Description
`mdsys.sdo_geometry`	The geometry for the polygon that approximates the circle.

Example

This example is a query that returns a polygon that approximates a circle at (0, 0) with radius 1.

```
select apex_spatial.circle_polygon(0, 0, 1) from dual
```

24.4 DELETE_GEOM_METADATA Procedure

This procedure deletes a spatial metadata record.

Syntax

```
APEX_SPATIAL.DELETE_GEOM_METADATA (
    p_table_name        IN VARCHAR2,
    p_column_name       IN VARCHAR2,
    p_drop_index        IN BOOLEAN DEFAULT FALSE );
```

Parameters

Table 24-4 DELETE_GEOM_METADATA Parameters

Parameter	Description
p_table_name	Name of the feature table.
p_column_name	Name of the column of type mdsys.sdo_geometry.
p_drop_index	If TRUE (default is FALSE), drop the spatial index on the column.

Example

This example deletes metadata on column CITIES.SHAPE and drops the spatial index on this column.

```
begin
    apex_spatial.delete_geom_metadata (
        p_table_name  => 'CITIES',
        p_column_name => 'SHAPE',
        p_drop_index  => true );
end;
```

24.5 INSERT_GEOM_METADATA Procedure

This procedure inserts a spatial metadata record and optionally creates a spatial index.

Syntax

```
APEX_SPATIAL.INSERT_GEOM_METADATA (
    p_table_name        IN VARCHAR2,
    p_column_name       IN VARCHAR2,
    p_diminfo           in mdsys.sdo_dim_array,
    p_srid              in t_srid,
    p_create_index_name IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 24-5 INSERT_GEOM_METADATA Parameters

Parameter	Description
p_table_name	The name of the feature table.
p_column_name	The name of the column of type mdsys.sdo_geometry.
p_diminfo	The SDO_DIM_ELEMENT array, ordered by dimension, with one entry for each dimension.
p_srid	The SRID value for the coordinate system for all geometries in the column.

Table 24-5 (Cont.) INSERT_GEOM_METADATA Parameters

Parameter	Description
p_create_index_name	If not null, a spatial index on the column is created with this name. Only simple column names are supported, function based indexes or indexes on object attributes cause an error. For more complex requirements, leave this parameter null (the default) and manually create the index.

Example

This example creates table CITIES, spatial metadata and an index on column CITIES.SHAPE.

```
create table cities (
    city_id   number primary key,
    city_name varchar2(30),
    shape     mdsys.sdo_geometry )
/
begin
    apex_spatial.insert_geom_metadata (
        p_table_name   => 'CITIES',
        p_column_name  => 'SHAPE',
        p_diminfo     => SDO_DIM_ARRAY (
            SDO_DIM_ELEMENT('X', -180, 180, 1),
            SDO_DIM_ELEMENT('Y',  -90,  90, 1) ),
        p_srid        => apex_spatial.c_wgs_84 );
end;
/
   create index cities_idx_shape on cities(shape) indextype is mdsys.spatial_index
/
```

24.6 INSERT_GEOM_METADATA_LONLAT Procedure

This procedure inserts a spatial metadata record that is suitable for longitude/latitude and optionally creates a spatial index.

Syntax

```
APEX_SPATIAL.INSERT_GEOM_METADATA_LONLAT (
    p_table_name        IN VARCHAR2,
    p_column_name       IN VARCHAR2,
    p_tolerance         IN NUMBER DEFAULT 1,
    p_create_index_name IN VARCHAR2 DEFAULT NULL );
```

Parameters

Table 24-6 INSERT_GEOM_METADATA_LONLAT Parameters

Parameter	Description
p_table_name	Name of the feature table.
p_column_name	Name of the column of type mdsys.sdo_geometry.

Table 24-6 (Cont.) INSERT_GEOM_METADATA_LONLAT Parameters

Parameter	Description
p_tolerance	Tolerance value in each dimension, in meters (default 1).
p_create_index_name	if not null, a spatial index on the column is created with this name. Only simple column names are supported, function based indexes or indexes on object attributes cause an error. For more complex requirements, leave this parameter null (the default) and manually create the index.

Example

The code below creates table CITIES and spatial metadata for the column CITIES.SHAPE. By passing CITIES_IDX_SHAPE to p_create_index_name, the API call automatically creates an index on the spatial column.

```
create table cities (
    city_id   number primary key,
    city_name varchar2(30),
    shape     mdsys.sdo_geometry )
/
begin
    apex_spatial.insert_geom_metadata_lonlat (
        p_table_name        => 'CITIES',
        p_column_name       => 'SHAPE',
        p_create_index_name => 'CITIES_IDX_SHAPE' );
end;
/
```

24.7 POINT Function

This function creates a point at (p_lon, p_lat).

Syntax

```
APEX_SPATIAL.POINT (
    p_lon        IN NUMBER,
    p_lat        IN NUMBER,
    p_srid       IN t_srid DEFAULT c_wgs_84 )
RETURN mdsys.sdo_geometry;
```

Parameters

Table 24-7 POINT parameters

Parameter	Description
p_lon	Longitude position.
p_lat	Latitude position.
p_srid	Reference system (default c_wgs_84).

Returns

Table 24-8 POINT Function Returns

Return	Description
`mdsys.sdo_geometry`	The geometry for the point.

Example

This example is a query that returns a point at (10, 50).

```
select apex_spatial.point(10, 50) from dual;
```

This example is equivalent to:

```
select mdsys.sdo_geometry(2001, 4326, sdo_point_type(10, 50, null), null, null) from
dual;
```

24.8 RECTANGLE Function

This function creates a rectangle from point at (`p_lon1`, `p_lat1`) to (`p_lon2`, `p_lat2`).

Syntax

```
APEX_SPATIAL.RECTANGLE (
    p_lon1        IN NUMBER,
    p_lat1        IN NUMBER,
    p_lon2        IN NUMBER,
    p_lat2        IN NUMBER,
    p_srid        IN t_srid DEFAULT c_wgs_84 )
RETURN mdsys.sdo_geometry;
```

Parameters

Table 24-9 RECTANGLE Parameters

Parameter	Description
`p_lon1`	Longitude position of the lower left point.
`p_lat1`	Latitude position of the lower left point.
`p_lon2`	Longitude position of the upper right point.
`p_lat2`	Latitude position of the upper right point.
`p_srid`	Reference system (default c_wgs_84).

Returns

Table 24-10 RECTANGLE Function Returns

Return	Description
`mdsys.sdo_geometry`	The geometry for the rectangle (p_lon1, p_lon2, p_lon2, p_lat2).

Example

This example is a query that returns a rectangle from (10, 50) to (11, 51).

```
select apex_spatial.rectangle(10, 50, 11, 51) from dual
```

This example is equivalent to:

```
select mdsys.sdo_geometry(
    2003, 4326, null,
    sdo_elem_info_array(1, 1003, 1),
    sdo_ordinate_array(10, 50, 11, 50, 11, 51, 10, 51, 10, 50))
from dual;
```

25

APEX_STRING

The APEX_STRING package provides utilities for `varchar2`, `clob`, `apex_t_varchar2`, and `apex_t_number` types.

25.1 FORMAT Function

Returns a formatted string, with substitutions applied.

Returns `p_message` after replacing each <n>th occurrence of `%s` with `p<n>` and each occurrence of `%<n>` with `p<n>`. If `p_max_length` is not null, `substr(p<n>, 1,p_arg_max_length)` is used instead of `p<n>`.

Syntax

```
format (
     p_message    in varchar2,
     p0           in varchar2     default null,
     p1           in varchar2     default null,
     p2           in varchar2     default null,
     p3           in varchar2     default null,
     p4           in varchar2     default null,
     p5           in varchar2     default null,
     p6           in varchar2     default null,
     p7           in varchar2     default null,
     p8           in varchar2     default null,
     p9           in varchar2     default null,
     p10          in varchar2     default null,
     p11          in varchar2     default null,
     p12          in varchar2     default null,
     p13          in varchar2     default null,
     p14          in varchar2     default null,
     p15          in varchar2     default null,
     p16          in varchar2     default null,
     p17          in varchar2     default null,
     p18          in varchar2     default null,
     p19          in varchar2     default null,
     p_max_length in pls_integer  default 1000 )
     return varchar2
```

Parameters

Table 25-1 FORMAT Function Parameters

Parameters	Description
`p_message`	Message string with substitution placeholders.
`p0-p19`	Substitution parameters.
`p_max_length`	If not null (default is 1000), cap each `p<n>` at `p_max_length` characters.

Example

```
apex_string.format('%s+%s=%s', 1, 2, 'three')
-> 1+2=three

apex_string.format('%1+%2=%0', 'three', 1, 2)
-> 1+2=three
```

25.2 GET_INITIALS Function

Get N letter initials from the first N words.

Syntax

```
get_initials (
   p_str in varchar2,
   p_cnt in number default 2 )
   return varchar2
```

Parameters

Table 25-2 GET_INITIALS Function Parameters

Parameters	Description
p_string	The input string.
p_cnt	The N letter initials to get from the first N words. The default is 2.

Example

Get initials from "John Doe".

```
begin
  sys.dbms_output.put_line(apex_string.get_initials('John Doe'));
end;
-> JD

begin
  sys.dbms_output.put_line(apex_string.get_initials(p_str => 'Andres Homero Lozano
Garza', p_cnt => 3));
end;
-> AHL
```

25.3 GREP Function Signature 1

Returns the values of the input table that match a regular expression.

Syntax

```
grep (
   p_table         in apex_t_varchar2,
   p_pattern       in varchar2,
   p_modifier      in varchar2    default null,
   p_subexpression in varchar2    default '0',
   p_limit         in pls_integer default null )
   return apex_t_varchar2;
```

Parameters

Table 25-3 GREP Function Signature 1 Parameters

Parameters	Description
p_table	The input table.
p_pattern	The regular expression.
p_modifier	The regular expression modifier.
p_subexpression	The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If > 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.

Table 25-3 (Cont.) GREP Function Signature 1 Parameters

Parameters	Description
p_limit	Limitation for the number of elements in the return table. If null (the default), there is no limit.

Example

Collect and print basenames of sql files in input collection.

```
declare
    l_sqlfiles apex_t_varchar2;
begin
    l_sqlfiles := apex_string.grep (
                       p_table => apex_t_varchar2('a.html','b.sql', 'C.SQL'),
                       p_pattern => '(\w+)\.sql',
                       p_modifier => 'i',
                       p_subexpression => '1' );
    sys.dbms_output.put_line(apex_string.join(l_sqlfiles, ':'));
end;
-> b:C
```

25.4 GREP Function Signature 2

Returns the values of the input `varchar2` that match a regular expression.

Syntax

```
grep (
   p_str           in varchar2,
   p_pattern       in varchar2,
   p_modifier      in varchar2    default null,
   p_subexpression in varchar2    default '0',
   p_limit         in pls_integer default null )
   return apex_t_varchar2;
```

Parameters

Table 25-4 GREP Function Signature 2 Parameters

Parameters	Description
p_str	The input `varchar2`.
p_pattern	The regular expression.
p_modifier	The regular expression modifier.
p_subexpression	The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If > 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.
p_limit	Limitation for the number of elements in the return table. If null (the default), there is no limit.

Example

Collect and print `key=value` definitions.

```
declare
    l_plist apex_t_varchar2;
begin
    l_plist := apex_string.grep (
                    p_str => 'define k1=v1'||chr(10)||
                             'define k2 = v2',
                    p_pattern => 'define\s+(\w+)\s*=\s*([^'||chr(10)||']*)',
                    p_modifier => 'i',
                    p_subexpression => '1,2' );
    sys.dbms_output.put_line(apex_string.join(l_plist, ':'));
end;
-> k1:v1:k2:v2
```

25.5 GREP Function Signature 3

Returns the values of the input `clob` that match a regular expression.

Syntax

```
grep (
   p_str           in clob,
   p_pattern       in varchar2,
   p_modifier      in varchar2    default null,
   p_subexpression in varchar2    default '0',
   p_limit         in pls_integer default null )
   return apex_t_varchar2;
```

Parameters

Table 25-5 GREP Function Signature 3 Parameters

Parameters	Description
`p_str`	The input `clob`.
`p_pattern`	The regular expression.
`p_modifier`	The regular expression modifier.
`p_subexpression`	The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If > 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.
`p_limit`	Limitation for the number of elements in the return table. If null (the default), there is no limit.

Example

Collect and print `key=value` definitions.

```
declare
    l_plist apex_t_varchar2;
begin
```

```
    l_plist := apex_string.grep (
                    p_str => to_clob('define k1=v1'||chr(10)||
                             'define k2 = v2',
                    p_pattern => 'define\s+(\w+)\s*=\s*([^'||
chr(10)||']*)',
                    p_modifier => 'i',
                    p_subexpression => '1,2' );
    sys.dbms_output.put_line(apex_string.join(l_plist, ':'));
end;
-> k1:v1:k2:v2
```

25.6 JOIN_CLOB Function

Returns the values of the `apex_t_varchar2` input table `p_table` as a concatenated `clob`, separated by `p_sep`.

Syntax

```
join_clob (
    p_table in apex_t_varchar2,
    p_sep   in varchar2    default apex_application.LF,
    p_dur   in pls_integer default sys.dbms_lob.call )
    return clob
```

Parameters

Table 25-6 JOIN_CLOB Function Parameters

Parameters	Description
`p_table`	The input table.
`p_sep`	The separator, default is line feed.
`p_dur`	The duration of the `clob`, default `sys.dbms_lob.call`.

Example

Concatenate numbers, separated by *':'*.

```
apex_string.join_clob(apex_t_varchar2(1,2,3),':')
-> 1:2:3
```

25.7 JOIN Function Signature 1

Returns the values of the `apex_t_varchar2` input table `p_table` as a concatenated `varchar2`, separated by `p_sep`.

Syntax

```
join (
   p_table in apex_t_varchar2,
   p_sep in varchar2 default apex_application.LF)
   return varchar2
```

Parameters

Table 25-7 JOIN Function Signature 1 Parameters

Parameters	Description
p_table	The input table.
p_sep	The separator, default is line feed.

Example

Concatenate numbers, separated by *':'*.

```
apex_string.join(apex_t_varchar2('a','b','c'),':')
-> a:b:c
```

25.8 JOIN Function Signature 2

Returns the values of the apex_t_number input table p_table as a concatenated varchar2, separated by p_sep.

Syntax

```
join (
   p_table in apex_t_number,
   p_sep in varchar2 default apex_application.LF )
   return varchar2
```

Parameters

Table 25-8 JOIN Function Signature 2 Parameters

Parameters	Description
p_table	The input table.
p_sep	The separator, default is line feed.

Example

Concatenate numbers, separated by *':'*.

```
apex_string.join(apex_t_number(1,2,3),':')
-> 1:2:3
```

25.9 PLIST_DELETE Procedure

This procedure removes the property list key from the table, by setting it to null.

Syntax

```
plist_delete (
     p_table in out nocopy apex_t_varchar2,
     p_key   in varchar2 );
```

Parameters

Table 25-9 PLIST_DELETE Procedure Parameters

Parameters	Description
p_table	The input table.
p_key	The input key.

Raised Errors

Table 25-10 PLIST_DELETE Procedure Raised Errors

Parameters	Description
NO_DATA_FOUND	Given key does not exist in table.

Example

Remove value of property"key2".

```
declare
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
begin
    apex_string.plist_delete(l_plist,'key2');
    sys.dbms_output.put_line(apex_string.join(l_plist,':'));
end;
-> key1:foo::
```

25.10 PLIST_GET Function

This function gets the property list value for a key.

Syntax

```
plist_get (
     p_table in apex_t_varchar2,
     p_key in varchar2 )
     return varchar2
```

Parameters

Table 25-11 PLIST_GET Function Parameters

Parameters	Description
p_table	The input table.
p_key	The input key.

Raised Errors

Table 25-12 PLIST_GET Function Raised Errors

Parameters	Description
NO_DATA_FOUND	Given key does not exist in table.

Example

Get value of property "key2".

```
declare
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
begin
    sys.dbms_output.put_line(apex_string.plist_get(l_plist,'key2'));
end;
-> bar
```

25.11 PLIST_PUT Function

This function inserts or updates property list value for a key.

Syntax

```
plist_put (
     p_table in out nocopy apex_t_varchar2,
     p_key   in varchar2,
     p_value in varchar2 );
```

Parameters

Table 25-13 PLIST_PUT Function Parameters

Parameters	Description
p_table	The input table.
p_key	The input key.
p_value	The input value.

Example

Set property value to "key2".

```
declare
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo');
begin
    apex_string.plist_put(l_plist,'key2','bar');
    sys.dbms_output.put_line(apex_string.plist_get(l_plist,'key2'));
end;
-> bar
```

25.12 PUSH Procedure Signature 1

This procedure appends value to apex_t_varchar2 table.

Syntax

```
push (
   p_table in out nocopy apex_t_varchar2,
   p_value in varchar2 );
```

Parameters

Table 25-14 PUSH Procedure Signature 1 Parameters

Parameter	Description
p_table	Defines the table.
p_src	Specifies the value to be added.

Example

The following example demonstrates how to append 2 values, then prints the table.

```
declare
    l_table apex_t_varchar2;
begin
    apex_string.push(l_table, 'a');
    apex_string.push(l_table, 'b');
    sys.dbms_output.put_line(apex_string.join(l_table, ':'));
end;
-> a:b
```

25.13 PUSH Procedure Signature 2

This procedure appends a value to `apex_t_number` table.

Syntax

```
push (
   p_table in out nocopy apex_t_number,
   p_value in number );
```

Parameters

Table 25-15 PUSH Procedure Signature 2 Parameters

Parameter	Description
p_table	Defines the table.
p_src	Specifies the value to be added.

Example

The following example demonstrates how to append 2 values, then prints the table.

```
declare
   l_table apex_t_number;
begin
   apex_string.push(l_table, 1);
   apex_string.push(l_table, 2);
```

```
   sys.dbms_output.put_line(apex_string.join(l_table, ':'));
end;
-> 1:2
```

25.14 PUSH Procedure Signature 3

This procedure appends collection values to `apex_t_varchar2` table.

Syntax

```
push (
   p_table  in out nocopy apex_t_varchar2,
   p_values in            apex_t_varchar2 );
```

Parameters

Table 25-16 PUSH Procedure Signature 3 Parameters

Parameter	Description
`p_table`	Defines the table.
`p_values`	Specifies the values that should be added to `p_table`.

Example

The following example demonstrates how to append a single value and multiple values, then prints the table.

```
declare
   l_table apex_t_varchar2;
begin
   apex_string.push(l_table, 'a');
   apex_string.push(l_table, apex_t_varchar2('1','2','3'));
   sys.dbms_output.put_line(apex_string.join(l_table, ':'));
end;
-> a:1:2:3
```

25.15 SHUFFLE Function

Returns the input table values, re-ordered.

Syntax

```
shuffle (
      p_table in apex_t_varchar2 )
      return apex_t_varchar2;
```

Parameters

Table 25-17 SHUFFLE Function Parameters

Parameters	Description
`p_table`	The input table.

Example

Shuffle and print `l_table`.

```
declare
    l_table apex_t_varchar2 := apex_string.split('1234567890',null);
begin
    sys.dbms_output.put_line(apex_string.join(apex_string.shuffle(l_table),':'));
end;
-> a permutation of 1:2:3:4:5:6:7:8:9:0
```

25.16 SHUFFLE Procedure

This procedure randomly re-orders the values of the input table.

Syntax

```
shuffle (
      p_table  in out nocopy apex_t_varchar2 );
```

Parameters

Table 25-18 SHUFFLE Procedure Parameters

Parameters	Description
`p_table`	The input table, which will be modified by the procedure.

Example

Shuffle and print `l_table`.

```
declare
    l_table apex_t_varchar2 := apex_string.split('1234567890',null);
begin
    apex_string.shuffle(l_table);
    sys.dbms_output.put_line(apex_string.join(l_table,':'));
end;
-> a permutation of 1:2:3:4:5:6:7:8:9:0
```

25.17 SPLIT Function Signature 1

Use this function to split input string at separator.

Syntax

```
split (
    p_str   in varchar2,
    p_sep   in varchar2 default apex_application.LF,
    p_limit in pls_integer default null )
    return apex_t_varchar2;
```

Parameters

Table 25-19 SPLIT Function Signature 1 Parameters

Parameters	Description
p_str	The input string.
p_sep	The separator. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression. The default is to split at line feed.
p_limit	Maximum number of splits, ignored if null. If smaller than the total possible number of splits, the last table element contains the rest.

Examples

```
apex_string.split(1||chr(10)||2||chr(10)||3)
-> apex_t_varchar2('1','2','3')

apex_string.split('1:2:3',':')
-> apex_t_varchar2('1','2','3')

apex_string.split('123',null)
-> apex_t_varchar2('1','2','3')

apex_string.split('1:2:3:4',':',2)
-> apex_t_varchar2('1','2:3:4')

apex_string.split('key1=val1, key2=val2','\s*[=,]\s*')
-> apex_t_varchar2('key1','val1','key2','val2')
```

25.18 SPLIT Function Signature 2

Use this function to split input clob at separator.

Syntax

```
split (
    p_str in clob,
    p_sep in varchar2 default apex_application.LF )
    return apex_t_varchar2;
```

Parameters

Table 25-20 SPLIT Function Signature 2 Parameters

Parameters	Description
p_str	The input clob.
p_sep	The separator. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression. The default is to split at line feed

Example

```
apex_string.split('1:2:3',':')
-> apex_t_varchar2('1','2','3')
```

25.19 SPLIT_NUMBERS Function

Use this function to split input at separator, values must all be numbers.

Syntax

```
split_numbers (
     p_str in varchar2,
     p_sep in varchar2 default apex_application.LF )
     return apex_t_number;
```

Parameters

Table 25-21 SPLIT_NUMBERS Function Parameters

Parameters	Description
p_str	The input varchar2.
p_sep	The separator. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression. The default is to split at line feed.

Example

```
apex_string.split_numbers('1:2:3',':')
-> apex_t_number(1,2,3)
```

26

APEX_THEME

The APEX_THEME package contains utility functions for working with themes and theme styles.

26.1 CLEAR_ALL_USERS_STYLE Procedure

This procedure clears all theme style user preferences for an application and theme.

Syntax

```
procedure clear_all_users_style(
    p_application_id  IN NUMBER           DEFAULT {current application id},
    p_theme_number    IN NUMBER           DEFAULT {current theme id}
);
```

Parameters

Table 26-1 CLEAR_ALL_USERS_STYLE Procedure

Parameter	Description
p_application_id	The application to clear all user theme style preferences for.
p_theme_number	The theme number to clear all theme style user preferences for.

Example

The following example clears the all theme style user preferences for theme 42 in application 100.

```
apex_theme.clear_all_users_style(
p_application_id => 100,
p_theme_number => 42
);
```

26.2 CLEAR_USER_STYLE Procedure

This procedure clears the theme style user preference for user and application.

Syntax

```
procedure clear_user_style(
    p_application_id IN NUMBER          DEFAULT  {current application id},
    p_user           IN VARCHAR2        DEFAULT {current user},
    p_theme_number   IN  NUMBER         DEFAULT {current theme number}
);
```

Parameters

Table 26-2 CLEAR_USER_STYLE Procedure

Parameter	Description
p_theme_number	The theme number to clear the theme style user preference.

Example

The following example clears the theme style user preference for the ADMIN user in application 100 and theme 42.

```
apex_theme.clear_user_style(
    p_application_id => 100,
    p_user           => 'ADMIN',
    p_theme_number   => 42
);
```

26.3 DISABLE_USER_STYLE Procedure

This procedure disables theme style selection by end users. End users will not be able to customize the theme style on their own. Note that this only affects the *Customization* link for end users. APEX_THEME API calls are independent.

Syntax

```
procedure disable_user_style(
    p_application_id  IN NUMBER           DEFAULT {current application id},
    p_theme_number    IN NUMBER           DEFAULT {current theme number}
);
```

Parameters

Table 26-3 DISABLE_USER_STYLE Procedure

Parameter	Description
p_application_id	The Application ID.

Table 26-3 (Cont.) DISABLE_USER_STYLE Procedure

Parameter	Description
p_theme_number	Number of User Interface's *Current Theme*.

The following example disable end user theme style selection for the `Desktop` user interface of application `100`.

```
declare
  l_theme_id apex_themes.theme_number%type;
begin
select theme_number into l_theme_id
  from apex_appl_user_interfaces
where application_id = 100
  and display_name   = 'Desktop';

apex_theme.disable_user_style(
  p_application_id => 100,
  p_theme_number   => l_theme_id
);
end;
```

26.4 ENABLE_USER_STYLE Procedure

This procedure enables theme style selection by end users. When enabled and there is at least one theme style marked as `Public`, end users will see a Customize link which allows to choose the theme style. End user theme style selection is enabled or disabled at the User Interface level. When providing a theme number, the theme must be the *Current Theme* for a user interface. Note that this only affects the *Customization* link for end users. `APEX_THEME` API calls are independent.

Syntax

```
procedure enable_user_style(
    p_application_id  IN NUMBER           DEFAULT {current application id},
    p_theme_number    IN NUMBER           DEFAULT {current theme number}
);
```

Parameters

Table 26-4 ENABLE_USER_STYLE Procedure

Parameter	Description
p_application_id	The Application ID.
p_theme_number	Number of User Interface's *Current Theme*.

The following example enable end user theme style selection for the `Desktop` user interface of application `100`.

```
declare
  l_theme_id apex_themes.theme_number%type;
begin
 select theme_number into l_theme_id
 from apex_appl_user_interfaces
```

```
 where application_id = 100
 and display_name   = 'Desktop';

 apex_theme.enable_user_style(
  p_application_id => 100,
  p_theme_number   => l_theme_id
);
end;
```

26.5 GET_USER_STYLE Function

This function returns the theme style user preference for the user and application. If no user preference is present, NULL is returned.

Syntax

```
function get_user_style(
    p_application_id  IN NUMBER           DEFAULT  {current application id},
    p_user            IN VARCHAR2         DEFAULT {current user},
    p_theme_number    IN NUMBER           DEFAULT {current theme number}
) return number;
```

Parameters

Table 26-5 GET_USER_STYLE Function

Parameter	Description
p_application_id	The application to set the user style preference.
p_user	The user name to the user style preference.
p_theme_number	The theme number to set the session style.
RETURN	The theme style ID which is set as a user preference.

Example

The query returns the theme style user preference for the ADMIN user in application 100 and theme 42.

```
select apex_theme.get_user_style( 100, 'ADMIN', 42 ) from dual;
```

26.6 SET_CURRENT_STYLE Procedure

This procedure sets current theme style for the current application.

Syntax

```
procedure set_current_style (
    p_theme_number  IN NUMBER,
    p_id            IN VARCHAR2
    );
```

Parameters

Table 26-6 SET_CURRENT_STYLE Procedure

Parameter	Description
p_theme_number	The theme number for which to set the default style.
p_style_id	The ID of the new default theme style.

Example

The following example gets available theme styles from **Application Express Dictionary View** for the DESKTOP user interface.

```
select s.theme_style_id, t.theme_number
  from apex_application_theme_styles s,
apex_application_themes t
          where s.application_id = t.application_id
            and s.theme_number = t.theme_number
            and s.application_id = :app_id
            and t.ui_type_name = 'DESKTOP'
            and s.is_current = 'Yes'
```

The following example sets the current theme style to one of values returned by the above query.

```
apex_theme.set_current_style (
     p_theme_number => {query.theme_number},
     p_id => {query.theme_style_id}
);
```

See Also:

"SET_CURRENT_THEME_STYLE Procedure [DEPRECATED] (page 28-96)"

26.7 SET_SESSION_STYLE Procedure

This procedure sets the theme style dynamically for the current session. This is typically being called after successful authentication.

Syntax

```
procedure set_session_style (
    p_theme_number  IN NUMBER           DEFAULT {current theme number},
    p_name          IN VARCHAR2
);
```

Parameters

Table 26-7 SET_SESSION_STYLE Procedure

Parameter	Description
p_theme_number	The theme number to set the session style for, default is the current theme of the application.
p_name	The name of the theme style to be used in the session.

Example

The following example gets the current theme number from **Application Express Dictionary View** for the DESKTOP user interface.

```
select t.theme_number
  from apex_application_themes t
 where t.application_id = :app_id
   and t.ui_type_name = 'DESKTOP'
```

The following example sets the session theme style for the current theme toVita.

```
apex_theme.set_current_style (
     p_theme_number => {query.theme_number},
     p_name => 'Vita'
);
```

26.8 SET_SESSION_STYLE_CSS Procedure

This procedure sets the theme style CSS urls dynamically for the current session. Theme style CSS URLs are being directly passed in; a persistent style definition is not needed. This is typically being called after successful authentication.

Syntax

```
procedure set_session_style_css (
    p_theme_number  IN NUMBER          DEFAULT {current theme number},
    p_css_file_urls IN VARCHAR2
);
```

Parameters

Table 26-8 SET_SESSION_STYLE_CSS Procedure

Parameter	Description
p_theme_number	The theme number to set the session style.
p_css_urls	The URLs to CSS files with style directives.

Example

The following example gets available theme styles from **Application Express Dictionary View** for the DESKTOP user interface.

```
select s.theme_style_id, t.theme_number
  from apex_application_theme_styles s,
apex_application_themes t
          where s.application_id = t.application_id
            and s.theme_number = t.theme_number
            and s.application_id = :app_id
            and t.ui_type_name = 'DESKTOP'
            and s.is_current = 'Yes'
```

The following example sets the current theme style to one of values returned by the above query.

```
apex_theme.set_session_style_css(
    p_theme_number => {query.theme_number},
    p_css_urls => {URLs to theme style CSS files}
);
```

26.9 SET_USER_STYLE Procedure

This procedure sets a theme style user preference for the current user and application. Theme Style User Preferences are automatically picked up and precede any style set with SET_SESSION_STYLE.

Syntax

```
procedure set_user_style(
    p_application_id  IN NUMBER          DEFAULT  {current application id},
    p_user            IN VARCHAR2        DEFAULT {current user},
    p_theme_number    IN NUMBER          DEFAULT {current theme number},
    p_id              IN NUMBER
);
```

Parameters

Table 26-9 SET_USER_STYLE Procedure

Parameter	Description
p_application_id	The application to set the user style preference.
p_user	The user name to the user style preference.
p_theme_number	The theme number to set the user style preference.
p_id	The ID of the theme style to set as a user preference.

Example

The following example gets available theme styles from **Application Express Dictionary View** for the DESKTOP user interface.

```
select s.theme_style_id, t.theme_number
  from apex_application_theme_styles s,
apex_application_themes t
          where s.application_id = t.application_id
            and s.theme_number = t.theme_number
```

```
            and s.application_id = :app_id
            and t.ui_type_name = 'DESKTOP'
            and s.is_current = 'Yes'
```

The following example sets the current theme style id's as user preference for `ADMIN` in application ID `100`.

```
apex_theme.set_user_style (
     p_application_id => 100,
     p_user           => 'ADMIN',
     p_theme_number   => {query.theme_number},
     p_id             => {query.theme_style_id}
);
```

27

APEX_UI_DEFAULT_UPDATE

The APEX_UI_DEFAULT_UPDATE package provides procedures to access user interface defaults from within SQL Developer or SQL*Plus.

You can use this package to set the user interface defaults associated with a table within a schema. The package must be called from within the schema that owns the table you are updating.

User interface defaults enable you to assign default user interface properties to a table, column, or view within a specified schema. When you create a form or report using a wizard, the wizard uses this information to create default values for region and item properties. Utilizing user interface defaults can save valuable development time and has the added benefit of providing consistency across multiple pages in an application.

UPD_LABEL Procedure (page 27-17)

UPD_REPORT_ALIGNMENT Procedure (page 27-18)

UPD_REPORT_FORMAT_MASK Procedure (page 27-18)

UPD_REPORT_REGION_TITLE Procedure (page 27-19)

UPD_TABLE Procedure (page 27-20)

See Also:

"Managing User Interface Defaults" in *Oracle Application Express SQL Workshop Guide*

27.1 ADD_AD_COLUMN Procedure

Adds a User Interface Default Attribute Dictionary entry with the provided definition. Up to three synonyms can be provided during the creation. Additional synonyms can be added post-creation using apex_ui_default_update.add_ad_synonym. Synonyms share the column definition of their base column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.ADD_AD_COLUMN (
    p_column_name           IN  VARCHAR2,
    p_label                 IN  VARCHAR2  DEFAULT NULL,
    p_help_text             IN  VARCHAR2  DEFAULT NULL,
    p_format_mask           IN  VARCHAR2  DEFAULT NULL,
    p_default_value         IN  VARCHAR2  DEFAULT NULL,
    p_form_format_mask      IN  VARCHAR2  DEFAULT NULL,
    p_form_display_width    IN  VARCHAR2  DEFAULT NULL,
    p_form_display_height   IN  VARCHAR2  DEFAULT NULL,
    p_form_data_type        IN  VARCHAR2  DEFAULT NULL,
    p_report_format_mask    IN  VARCHAR2  DEFAULT NULL,
    p_report_col_alignment  IN  VARCHAR2  DEFAULT NULL,
    p_syn_name1             IN  VARCHAR2  DEFAULT NULL,
    p_syn_name2             IN  VARCHAR2  DEFAULT NULL,
    p_syn_name3             IN  VARCHAR2  DEFAULT NULL);
```

Parameters

Table 27-1 ADD_AD_COLUMN Parameters

Parameter	Description
p_column_name	Name of column to be created.
p_label	Used for item label and report column heading.
p_help_text	Used for help text for items and interactive report columns
p_format_mask	Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.
p_default_value	Used as the default value for items.

Table 27-1 (Cont.) ADD_AD_COLUMN Parameters

Parameter	Description
p_form_format_mask	If provided, used as the format mask for items, overriding any value for the general format mask.
p_form_display_width	Used as the width of any items using this Attribute Definition.
p_form_display_height	Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).
p_form_data_type	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
p_report_format_mask	If provided, used as the format mask for report columns, overriding any value for the general format mask.
p_report_col_alignment	Used as the alignment for report column data (for example, number are usually right justified). Valid values are LEFT, CENTER, and RIGHT.
p_syn_name1	Name of synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.
p_syn_name2	Name of second synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.
p_syn_name3	Name of third synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.

Example

The following example creates a new attribute to the UI Defaults Attribute Dictionary within the workspace associated with the current schema. It also creates a synonym for that attribute.

```
BEGIN
    apex_ui_default_update.add_ad_column (
       p_column_name          => 'CREATED_BY',
       p_label                => 'Created By',
       p_help_text            => 'User that created the record.',
       p_form_display_width   => 30,
       p_form_data_type       => 'VARCHAR',
       p_report_col_alignment => 'LEFT',
       p_syn_name1            => 'CREATED_BY_USER' );
END;
```

27.2 ADD_AD_SYNONYM Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the synonym provided is created and associated with that column. Synonyms share the column definition of their base column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM (
    p_column_name           IN VARCHAR2,
    p_syn_name              IN VARCHAR2);
```

Parameters

Table 27-2 ADD_AD_SYNONYM Parameters

Parameter	Description
p_column_name	Name of column with the Attribute Dictionary that the synonym is being created for.
p_syn_name	Name of synonym to be created.

Example

The following example add the synonym CREATED_BY_USER to the CREATED_BY attribute of the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.add_ad_synonym (
       p_column_name => 'CREATED_BY',
       p_syn_name    => 'CREATED_BY_USER' );
END;
```

27.3 DEL_AD_COLUMN Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the column, along with any associated synonyms, is deleted.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_AD_COLUMN (
    p_column_name           IN VARCHAR2);
```

Parameters

Table 27-3 DEL_AD_COLUMN Parameters

Parameter	Description
p_column_name	Name of column to be deleted

Example

The following example deletes the attribute CREATED_BY from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_column (
       p_column_name => 'CREATED_BY' );
END;
```

27.4 DEL_AD_SYNONYM Procedure

If the synonym name is found within the User Interface Default Attribute Dictionary, the synonym name is deleted.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_AD_SYNONYM (
    p_syn_name           IN VARCHAR2);
```

Parameters

Table 27-4 DEL_AD_SYNONYM Parameters

Parameter	Description
p_syn_name	Name of synonym to be deleted

Example

The following example deletes the synonym CREATED_BY_USER from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_synonym (
       p_syn_name    => 'CREATED_BY_USER' );
END;
```

27.5 DEL_COLUMN Procedure

If the provided table and column exists within the user's schema's table based User Interface Defaults, the UI Defaults for it are deleted.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_COLUMN (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2);
```

Parameters

Table 27-5 DEL_COLUMN Parameters

Parameter	Description
p_table_name	Name of table whose column's UI Defaults are to be deleted.

Table 27-5 (Cont.) DEL_COLUMN Parameters

Parameter	Description
p_column_name	Name of columns whose UI Defaults are to be deleted.

Example

The following example deletes the column CREATED_BY from the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.del_column (
       p_table_name  => 'EMP',
       p_column_name => 'CREATED_BY' );
END;
```

27.6 DEL_GROUP Procedure

If the provided table and group exists within the user's schema's table based User Interface Defaults, the UI Defaults for it are deleted and any column within the table that references that group has the group_id set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_GROUP (
    p_table_name            IN VARCHAR2,
    p_group_name            IN VARCHAR2);
```

Parameters

Table 27-6 DEL_GROUP Parameters

Parameter	Description
p_table_name	Name of table whose group UI Defaults are to be deleted
p_group_name	Name of group whose UI Defaults are to be deleted

Example

The following example deletes the group AUDIT_INFO from the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.del_group (
       p_table_name => 'EMP',
       p_group_name => 'AUDIT_INFO' );
END;
```

27.7 DEL_TABLE Procedure

If the provided table exists within the user's schema's table based User Interface Defaults, the UI Defaults for it is deleted. This includes the deletion of any groups defined for the table and all the columns associated with the table.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_TABLE (
    p_table_name            IN VARCHAR2);
```

Parameters

Table 27-7 DEL_TABLE Parameters

Parameter	Description
p_table_name	Table name

Example

The following example removes the UI Defaults for the EMP table that are associated with the current schema.

```
begin
    apex_ui_default_update.del_table (
       p_table_name => 'EMP' );
end;
/
```

27.8 SYNCH_TABLE Procedure

If the Table Based User Interface Defaults for the table do not already exist within the user's schema, they are defaulted. If they do exist, they are synchronized, meaning, the columns in the table is matched against the column in the UI Defaults Table Definitions. Additions and deletions are used to make them match.

Syntax

```
APEX_UI_DEFAULT_UPDATE.SYNCH_TABLE (
    p_table_name            IN VARCHAR2);
```

Parameters

Table 27-8 SYNCH_TABLE Parameters

Parameter	Description
p_table_name	Table name

Example

The following example synchronizes the UI Defaults for the EMP table that are associated with the current schema.

```
BEGIN
    apex_ui_default_update.synch_table (
       p_table_name => 'EMP' );
END;
```

27.9 UPD_AD_COLUMN Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the column entry is updated using the provided parameters. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_AD_COLUMN (
    p_column_name           IN  VARCHAR2,
    p_new_column_name       IN  VARCHAR2  DEFAULT NULL,
    p_label                 IN  VARCHAR2  DEFAULT NULL,
    p_help_text             IN  VARCHAR2  DEFAULT NULL,
    p_format_mask           IN  VARCHAR2  DEFAULT NULL,
    p_default_value         IN  VARCHAR2  DEFAULT NULL,
    p_form_format_mask      IN  VARCHAR2  DEFAULT NULL,
    p_form_display_width    IN  VARCHAR2  DEFAULT NULL,
    p_form_display_height   IN  VARCHAR2  DEFAULT NULL,
    p_form_data_type        IN  VARCHAR2  DEFAULT NULL,
    p_report_format_mask    IN  VARCHAR2  DEFAULT NULL,
    p_report_col_alignment  IN  VARCHAR2  DEFAULT NULL);
```

Parameters

Table 27-9 UPD_AD_COLUMN Parameters

Parameter	Description
p_column_name	Name of column to be updated
p_new_column_name	New name for column, if column is being renamed
p_label	Used for item label and report column heading
p_help_text	Used for help text for items and interactive report columns
p_format_mask	Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.
p_default_value	Used as the default value for items.
p_form_format_mask	If provided, used as the format mask for items, overriding any value for the general format mask.
p_form_display_width	Used as the width of any items using this Attribute Definition.
p_form_display_height	Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).
p_form_data_type	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
p_report_format_mask	If provided, used as the format mask for report columns, overriding any value for the general format mask.

Table 27-9 (Cont.) UPD_AD_COLUMN Parameters

Parameter	Description
p_report_col_alignment	Used as the alignment for report column data (for example, number are usually right justified). Valid values are LEFT, CENTER, and RIGHT.

Note:

If p_label through p_report_col_alignment are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the CREATED_BY column in the UI Defaults Attribute Dictionary within the workspace associated with the current schema, setting the form_format_mask to null.

```
BEGIN
    apex_ui_default_update.upd_ad_column (
       p_column_name      => 'CREATED_BY',
       p_form_format_mask => 'null%');
END;
```

27.10 UPD_AD_SYNONYM Procedure

If the synonym name is found within the User Interface Default Attribute Dictionary, the synonym name is updated.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_AD_SYNONYM (
    p_syn_name           IN VARCHAR2,
    p_new_syn_name       IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-10 UPD_AD_SYNONYM Parameters

Parameter	Description
p_syn_name	Name of synonym to be updated
p_new_syn_name	New name for synonym

Example

The following example updates the CREATED_BY_USER synonym in the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.upd_ad_synonym (
       p_syn_name     => 'CREATED_BY_USER',
```

```
      p_new_syn_name => 'USER_CREATED_BY');
END;
```

27.11 UPD_COLUMN Procedure

If the provided table and column exists within the user's schema's table based User Interface Defaults, the provided parameters are updated. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_COLUMN (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_group_id              IN VARCHAR2  DEFAULT NULL,
    p_label                 IN VARCHAR2  DEFAULT NULL,
    p_help_text             IN VARCHAR2  DEFAULT NULL,
    p_display_in_form       IN VARCHAR2  DEFAULT NULL,
    p_display_seq_form      IN VARCHAR2  DEFAULT NULL,
    p_mask_form             IN VARCHAR2  DEFAULT NULL,
    p_default_value         IN VARCHAR2  DEFAULT NULL,
    p_required              IN VARCHAR2  DEFAULT NULL,
    p_display_width         IN VARCHAR2  DEFAULT NULL,
    p_max_width             IN VARCHAR2  DEFAULT NULL,
    p_height                IN VARCHAR2  DEFAULT NULL,
    p_display_in_report     IN VARCHAR2  DEFAULT NULL,
    p_display_seq_report    IN VARCHAR2  DEFAULT NULL,
    p_mask_report           IN VARCHAR2  DEFAULT NULL,
    p_alignment             IN VARCHAR2  DEFAULT NULL);
```

Parameters

Table 27-11 UPD_COLUMN Parameters

Parameter	Description
p_table_name	Name of table whose column's UI Defaults are being updated
p_column_name	Name of column whose UI Defaults are being updated
p_group_id	id of group to be associated with the column
p_label	When creating a form against this table or view, this is used as the label for the item if this column is included. When creating a report or tabular form, this is used as the column heading if this column is included.
p_help_text	When creating a form against this table or view, this becomes the help text for the resulting item.
p_display_in_form	When creating a form against this table or view, this determines whether this column is displayed in the resulting form page. Valid values are Y and N.
p_display_seq_form	When creating a form against this table or view, this determines the sequence in which the columns is displayed in the resulting form page.

Table 27-11 (Cont.) UPD_COLUMN Parameters

Parameter	Description
p_mask_form	When creating a form against this table or view, this specifies the mask that is applied to the item, such as 999-99-9999. This is not used for character based items.
p_default_value	When creating a form against this table or view, this specifies the default value for the item resulting from this column.
p_required	When creating a form against this table or view, this specifies to generate a validation in which the resulting item must be NOT NULL. Valid values are Y and N.
p_display_width	When creating a form against this table or view, this specifies the display width of the item resulting from this column.
p_max_width	When creating a form against this table or view, this specifies the maximum string length that a user is allowed to enter in the item resulting from this column.
p_height	When creating a form against this table or view, this specifies the display height of the item resulting from this column.
p_display_in_report	When creating a report against this table or view, this determines whether this column is displayed in the resulting report. Valid values are Y and N.
p_display_seq_report	When creating a report against this table or view, this determines the sequence in which the columns are displayed in the resulting report.
p_mask_report	When creating a report against this table or view, this specifies the mask that is applied against the data, such as 999-99-9999. This is not used for character based items.
p_alignment	When creating a report against this table or view, this determines the alignment for the resulting report column. Valid values are L for Left, C for Center, and R for Right.

Note:

If p_group_id through p_alignment are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the column DEPT_NO within the EMP table definition within the UI Defaults Table Dictionary within the current schema, setting the group_id to null.

```
BEGIN
    apex_ui_default_update.upd_column (
       p_table_name    => 'EMP',
```

```
      p_column_name   => 'DEPT_NO',
      p_group_id      => 'null%' );
END;
```

27.12 UPD_DISPLAY_IN_FORM Procedure

The UPD_DISPLAY_IN_FORM procedure sets the display in form user interface defaults. This user interface default is used by wizards when you select to create a form based upon the table. It controls whether the column is included by default or not.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_in_form       IN VARCHAR2);
```

Parameters

Table 27-12 UPD_DISPLAY_IN_FORM Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_in_form	Determines whether to display in the form by default, valid values are Y and N

Example

In the following example, when creating a Form against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM(
    p_table_name => 'DEPT',
    p_column_name => 'DEPTNO',
    p_display_in_form => 'N');
```

27.13 UPD_DISPLAY_IN_REPORT Procedure

The UPD_DISPLAY_IN_REPORT procedure sets the display in report user interface default. This user interface default is used by wizards when you select to create a report based upon the table and controls whether the column is included by default or not.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_REPORT (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_in_report     IN VARCHAR2);
```

Parameters

Table 27-13 UPD_DISPLAY_IN_REPORT Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_in_report	Determines whether to display in the report by default, valid values are Y and N

Example

In the following example, when creating a Report against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_REPORT(
    p_table_name => 'DEPT',
    p_column_name => 'DEPTNO',
    p_display_in_report => 'N');
```

27.14 UPD_FORM_REGION_TITLE Procedure

The UPD_FORM_REGION_TITLE procedure updates the Form Region Title user interface default. User interface defaults are used in wizards when you create a form based upon the specified table.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_FORM_REGION_TITLE (
    p_table_name            IN VARCHAR2,
    p_form_region_title     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-14 UPDATE_FORM_REGION_TITLE Parameters

Parameter	Description
p_table_name	Table name
p_form_region_title	Desired form region title

Example

This example demonstrates how to set the Forms Region Title user interface default on the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_FORM_REGION_TITLE (
    p_table_name         => 'DEPT',
    p_form_region_title  => 'Deptartment Details');
```

27.15 UPD_GROUP Procedure

If the provided table and group exist within the user's schema's table based User Interface Defaults, the group name, description and display sequence of the group are updated. If 'null%' is passed in for p_description or p_display_sequence, the value is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_GROUP (
    p_table_name            IN VARCHAR2,
    p_group_name            IN VARCHAR2,
    p_new_group_name        IN VARCHAR2 DEFAULT NULL,
    p_description           IN VARCHAR2 DEFAULT NULL,
    p_display_sequence      IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-15 UPD_GROUP Parameters

Parameter	Description
p_table_name	Name of table whose group is being updated
p_group_name	Group being updated
p_new_group_name	New name for group, if group is being renamed
p_description	Description of group
p_display_sequence	Display sequence of group.

Note:

If p_description or p_display_sequence are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the description of the group AUDIT_INFO within the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.upd_group (
       p_table_name  => 'EMP',
       p_group_name  => 'AUDIT_INFO',
       p_description => 'Audit columns' );
END;
```

27.16 UPD_ITEM_DISPLAY_HEIGHT Procedure

The UPD_ITEM_DISPLAY_HEIGHT procedure sets the item display height user interface default. This user interface default is used by wizards when you select to

create a form based upon the table and include the specified column. Display height controls if the item is a text box or a text area.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_height        IN NUMBER);
```

Parameters

Table 27-16 UPD_ITEM_DISPLAY_HEIGHT Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_height	Display height of any items created based upon this column

Example

The following example sets a default item height of 3 when creating an item on the DNAME column against the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT(
   p_table_name => 'DEPT',
   p_column_name => 'DNAME',
   p_display_height => 3);
```

27.17 UPD_ITEM_DISPLAY_WIDTH Procedure

The UPD_ITEM_DISPLAY_WIDTH procedure sets the item display width user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_WIDTH (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_width         IN NUMBER);
```

Parameters

Table 27-17 UPD_ITEM_DISPLAY_WIDTH Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_width	Display width of any items created based upon this column

Example

The following example sets a default item width of 5 when creating an item on the DEPTNO column against the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_WIDTH(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_display_width => 5);
```

27.18 UPD_ITEM_FORMAT_MASK Procedure

The UPD_ITEM_FORMAT_MASK procedure sets the item format mask user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column. Item format mask is typically used to format numbers and dates.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_FORMAT_MASK (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_format_mask           IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-18 UPD_ITEM_FORMAT_MASK Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_format_mask	Format mask to be associated with the column

Example

In the following example, when creating a Form against the EMP table, the default item format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_FORMAT_MASK(
    p_table_name => 'EMP',
    p_column_name => 'HIREDATE',
    p_format_mask=> 'DD-MON-YYYY');
```

27.19 UPD_ITEM_HELP Procedure

The UPD_ITEM_HELP procedure updates the help text for the specified table and column. This user interface default is used when you create a form based upon the table and select to include the specified column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_help_text             IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-19 UPD_ITEM_HELP Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_help_text	Desired help text

Example

This example demonstrates how to set the User Interface Item Help Text default for the DEPTNO column in the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_help_text => 'The number assigned to the department.');
```

27.20 UPD_LABEL Procedure

The UPD_LABEL procedure sets the label used for items. This user interface default is used when you create a form or report based on the specified table and include a specific column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_LABEL (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_label                 IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-20 UPD__LABEL Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_label	Desired item label

Example

This example demonstrates how to set the User Interface Item Label default for the DEPTNO column in the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_LABEL(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_label => 'Department Number');
```

27.21 UPD_REPORT_ALIGNMENT Procedure

The UPD_REPORT_ALIGNMENT procedure sets the report alignment user interface default. This user interface default is used by wizards when you select to create a report based upon the table and include the specified column and determines if the report column should be left, center, or right justified.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_ALIGNMENT (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_report_alignment      IN VARCHAR2);
```

Parameters

Table 27-21 UPD_REPORT_ALIGNMENT Parameters

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_report_alignment	Defines the alignment of the column in a report. Valid values are L (left), C (center) and R (right).

Example

In the following example, when creating a Report against the DEPT table, the default column alignment on the DEPTNO column is set to Right justified.

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_ALIGNMENT(
    p_table_name => 'DEPT',
    p_column_name => 'DEPTNO',
    p_report_alignment => 'R');
```

27.22 UPD_REPORT_FORMAT_MASK Procedure

The UPD_REPORT_FORMAT_MASK procedure sets the report format mask user interface default. This user interface default is used by wizards when you select to create a report based upon the table and include the specified column. Report format mask is typically used to format numbers and dates.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_FORMAT_MASK (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_format_mask           IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-22 UPD_REPORT_FORMAT_MASK Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_format_mask	Format mask to be associated with the column whenever it is included in a report

Example

In the following example, when creating a Report against the EMP table, the default format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_FORMAT_MASK(
    p_table_name => 'EMP',
    p_column_name => 'HIREDATE',
    p_format_mask=> 'DD-MON-YYYY');
```

27.23 UPD_REPORT_REGION_TITLE Procedure

The UPD_REPORT_REGION_TITLE procedure sets the Report Region Title. User interface defaults are used in wizards when a report is created on a table.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_REGION_TITLE (
    p_table_name            IN VARCHAR2,
    p_report_region_title   IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-23 UPD_REPORT_REGION_TITLE Parameters

Parameter	Description
p_table_name	Table name
p_report_region_title	Desired report region title

Example

This example demonstrates how to set the Reports Region Title user interface default on the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_REGION_TITLE (
    p_table_name            => 'DEPT',
    p_report_region_title   => 'Departments');
```

27.24 UPD_TABLE Procedure

If the provided table exists within the user's schema's table based User Interface Defaults, the form region title and report region title are updated to match those provided. If 'null%' is passed in for p_form_region_title or p_report_region_title, the value is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_TABLE (
    p_table_name            IN VARCHAR2,
    p_form_region_title     IN VARCHAR2 DEFAULT NULL,
    p_report_region_title   IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-24 UPD_TABLE Parameters

Parameter	Description
`p_table_name`	Name of table being updated.
`p_form_region_title`	Region title used for forms.
p_report_region_title	Region title used for reports and tabular forms.

Note:

if 'null%' is passed in for `p_form_region_title` or `p_report_region_title`, the value is set to `null`. If no value is passed in, that column is not updated.

Example

The following example updates the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
begin
    apex_ui_default_update.upd_table (
       p_table_name          => 'EMP',
       p_form_region_title   => 'Employee Details',
       p_report_region_title => 'Employees' );
end;
/
```

28

APEX_UTIL

The APEX_UTIL package provides utilities you can use when programming in the Oracle Application Express environment. You can use the APEX_UTIL package to get and set session state, get files, check authorizations for users, reset different states for users, get and purge cache information and also to get and set preferences for users.

28.1 CACHE_GET_DATE_OF_PAGE_CACHE Function

This function returns the date and time a specified application page was cached either for the user issuing the call, or for all users if the page was not set to be cached by user.

Syntax

```
APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE (
    p_application  IN NUMBER,
    p_page         IN NUMBER)
RETURN DATE;
```

Parameters

Table 28-1 CACHE_GET_DATE_OF_PAGE_CACHE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The page number (ID).

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_PAGE_CACHE function to retrieve the cache date and time for page 9 of the currently executing application. If page 9 has been cached, the cache date and time is output using the HTP package. The page could have been cached either by the user issuing the call, or for all users if the page was not to be cached by the user.

```
DECLARE
    l_cache_date DATE DEFAULT NULL;
BEGIN
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE(
                        p_application => :APP_ID,
                        p_page => 9);
    IF l_cache_date IS NOT NULL THEN
        HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
    END IF;
END;
```

28.2 CACHE_GET_DATE_OF_REGION_CACHE Function

This function returns the date and time a specified region was cached either for the user issuing the call, or for all users if the page was not set to be cached by user.

Syntax

```
APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE (
    p_application  IN NUMBER,
    p_page         IN NUMBER,
    p_region_name  IN VARCHAR2)
RETURN DATE;
```

Parameters

Table 28-2 CACHE_GET_DATE_OF_REGION_CACHE Parameters

Parameter	Description
p_application	The identification number (ID) of the application .
p_page	The page number (ID).
p_region_name	The region name.

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_REGION_CACHE function to retrieve the cache date and time for the region named Cached Region on page 13 of the currently executing application. If the region has been cached, the cache date and time is output using the HTP package. The region could have been cached either by the user issuing the call, or for all users if the page was not to be cached by user.

```
DECLARE
    l_cache_date DATE DEFAULT NULL;
BEGIN
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE(
        p_application => :APP_ID,
        p_page => 13,
        p_region_name => 'Cached Region');
    IF l_cache_date IS NOT NULL THEN
        HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
    END IF;
END;
```

28.3 CACHE_PURGE_BY_APPLICATION Procedure

This procedure purges all cached pages and regions for a given application.

Syntax

```
APEX_UTIL.CACHE_PURGE_BY_APPLICATION (
    p_application  IN NUMBER);
```

Parameters

Table 28-3 CACHE_PURGE_BY_APPLICATION Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

The following example demonstrates how to use the CACHE_PURGE_BY_APPLICATION procedure to purge all the cached pages and regions for the application currently executing.

```
BEGIN
    APEX_UTIL.CACHE_PURGE_BY_APPLICATION(p_application => :APP_ID);
END;
```

28.4 CACHE_PURGE_BY_PAGE Procedure

This procedure purges the cache for a given application and page. If the page itself is not cached but contains one or more cached regions, then the cache for these is also purged.

Syntax

```
APEX_UTIL.CACHE_PURGE_BY_PAGE (
    p_application  IN NUMBER,
```

```
    p_page         IN NUMBER,
    p_user_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-4 CACHE_PURGE_BY_PAGE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The page number (ID).
p_user_name	The user associated with cached pages and regions.

Example

The following example demonstrates how to use the CACHE_PURGE_BY_PAGE procedure to purge the cache for page 9 of the application currently executing. Additionally, if the p_user_name parameter is supplied, this procedure would be further restricted by a specific users cache (only relevant if the cache is set to be by user).

```
BEGIN
    APEX_UTIL.CACHE_PURGE_BY_PAGE(
        p_application => :APP_ID,
        p_page => 9);
END;
```

28.5 CACHE_PURGE_STALE Procedure

This procedure deletes all cached pages and regions for a specified application that have passed the defined active time period. When you cache a page or region, you specify an active time period (or Cache Timeout). Once that period has passed, the cache is no longer used, thus removing those unusable pages or regions from the cache.

Syntax

```
APEX_UTIL.CACHE_PURGE_STALE (
    p_application  IN NUMBER);
```

Parameters

Table 28-5 CACHE_PURGE_STALE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

The following example demonstrates how to use the CACHE_PURGE_STALE procedure to purge all the stale pages and regions in the application currently executing.

```
BEGIN
    APEX_UTIL.CACHE_PURGE_STALE(p_application => :APP_ID);
END;
```

28.6 CHANGE_CURRENT_USER_PW Procedure

This procedure changes the password of the currently authenticated user, assuming Application Express user accounts are in use.

Syntax

```
APEX_UTIL.CHANGE_CURRENT_USER_PW(
    p_new_password IN VARCHAR2);
```

Parameters

Table 28-6 CHANGE_CURRENT_USER_PW Parameters

Parameter	Description
p_new_password	The new password value in clear text.

Example

The following example demonstrates how to use the CHANGE_CURRENT_USER_PW procedure to change the password for the user who is currently authenticated, assuming Application Express accounts are in use.

```
BEGIN
    APEX_UTIL.CHANGE_CURRENT_USER_PW ('secret99');
END;
```

See Also:

"RESET_PW Procedure (page 28-89)"

28.7 CHANGE_PASSWORD_ON_FIRST_USE Function

Enables a developer to check whether this property is enabled or disabled for an end user account. This function returns TRUE if the account password must be changed upon first use (after successful authentication) after the password is initially set and after it is changed on the Administration Service, Edit User page. This function returns FALSE if the account does not have this property.

This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE (
    p_user_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 28-7 CHANGE_PASSWORD_ON_FIRST_USE Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example demonstrates how to use the CHANGE_PASSWORD_ON_FIRST_USE function. Use this function to check if the password of an Application Express user account (workspace administrator, developer, or end user) in the current workspace must be changed by the user the first time it is used.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) LOOP
        IF APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE(p_user_name => c1.user_name) THEN
            htp.p('User:'||c1.user_name||' requires password to be changed the first
time it is used.');
        END IF;
    END LOOP;
END;
```

See Also:

"PASSWORD_FIRST_USE_OCCURRED Function (page 28-80)"

28.8 CLOSE_OPEN_DB_LINKS Procedure

This procedure closes all open database links for the current database session. It is rare that this procedure would ever be called programatically in an application. The primary purpose of this procedure is for the middleware technology in an Oracle Application Express environment (for example, Oracle REST Data Service, mod_plsql) to be configured such that it closes all of the open database links in a session, either before a request is made to the Application Express engine, or after a request to the Application Express engine is completed but before the database session is returned to the pool.

Syntax

```
APEX_UTIL.CLOSE_OPEN_DB_LINKS
```

Parameters

None.

Example

In this example, the configuration of Oracle REST Data Services closes any open database links both before the request is made to the Application Express engine and after the request is complete.

```
<entry key="procedure.postProcess">apex_util.close_open_db_links</entry>
<entry key="procedure.preProcess">apex_util.close_open_db_links</entry>
```

When using Oracle HTTP Server and `mod_plsql`, this configuration would look like this:

```
PlsqlBeforeProcedure    apex_util.close_open_db_links
PlsqlAfterProcedure        apex_util.close_open_db_links
```

28.9 CLEAR_APP_CACHE Procedure

This procedure removes session state for a given application for the current session.

Syntax

```
APEX_UTIL.CLEAR_APP_CACHE (
    p_app_id    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-8 CLEAR_APP_CACHE Parameters

Parameter	Description
p_app_id	The ID of the application for which session state is cleared for current session.

Example

The following example demonstrates how to use the CLEAR_APP_CACHE procedure to clear all the current sessions state for the application with an ID of 100.

```
BEGIN
    APEX_UTIL.CLEAR_APP_CACHE('100');
END;
```

28.10 CLEAR_PAGE_CACHE Procedure

This procedure removes session state for a given page for the current session.

Syntax

```
APEX_UTIL.CLEAR_PAGE_CACHE (
    p_page_id IN NUMBER DEFAULT NULL);
```

Parameters

Table 28-9 CLEAR_PAGE_CACHE Parameters

Parameter	Description
p_page_id	The ID of the page in the current application for which session state is cleared for current session.

Example

The following example demonstrates how to use the CLEAR_PAGE_CACHE procedure to clear the current session state for the page with an ID of 10.

```
BEGIN
    APEX_UTIL.CLEAR_PAGE_CACHE(10);
END;
```

28.11 CLEAR_USER_CACHE Procedure

This procedure removes session state and application system preferences for the current user's session. Run this procedure if you reuse session IDs and want to run applications without the benefit of existing session state.

Syntax

```
APEX_UTIL.CLEAR_USER_CACHE;
```

Parameters

None.

Example

The following example demonstrates how to use the CLEAR_USER_CACHE procedure to clear all session state and application system preferences for the current user's session.

```
BEGIN
    APEX_UTIL.CLEAR_USER_CACHE;
END;
```

28.12 COUNT_CLICK Procedure

This procedure counts clicks from an application built in App Builder to an external site. You can also use the shorthand version, procedure `Z`, in place of `APEX_UTIL.COUNT_CLICK`.

Syntax

```
APEX_UTIL.COUNT_CLICK (
    p_url         IN    VARCHAR2,
    p_cat         IN    VARCHAR2,
    p_id          IN    VARCHAR2    DEFAULT NULL,
    p_user        IN    VARCHAR2    DEFAULT NULL,
    p_workspace   IN    VARCHAR2    DEFAULT NULL);
```

Parameters

Table 28-10 COUNT_CLICK Parameters

Parameter	Description
`p_url`	The URL to which to redirect
`p_cat`	A category to classify the click
`p_id`	Secondary ID to associate with the click (optional)
`p_user`	The application user ID (optional)

Table 28-10 (Cont.) COUNT_CLICK Parameters

Parameter	Description
p_workspace	The workspace associated with the application (optional)

Example

The following example demonstrates how to use the COUNT_CLICK procedure to log how many user's click on the `http://yahoo.com` link specified. Note that once this information is logged, you can view it by using the APEX_WORKSPACE_CLICKS view and in the reports on this view available to workspace and site administrators.

```
DECLARE
    l_url VARCHAR2(255);
    l_cat VARCHAR2(30);
    l_workspace_id VARCHAR2(30);
BEGIN
    l_url := 'http://yahoo.com';
    l_cat := 'yahoo';
    l_workspace_id := TO_CHAR(APEX_UTIL.FIND_SECURITY_GROUP_ID('MY_WORKSPACE'));

    HTP.P('<a href=APEX_UTIL.COUNT_CLICK?p_url=' || l_url || '&p_cat=' || l_cat ||
'&p_workspace=' || l_workspace_id || '>Click</a>');
END;
```

See Also:

- "FIND_SECURITY_GROUP_ID Function (page 28-40)"in this document and "Deleting Click Counting Log Entries" in *Oracle Application Express Administration Guide*
- "Managing Authorized URLs" in *Oracle Application Express Administration Guide*

28.13 CREATE_USER Procedure

This procedure creates a new account record in the Application Express user account table. To execute this procedure, the current user must have administrative privileges.

Syntax

```
APEX_UTIL.CREATE_USER(
    p_user_id                      IN      NUMBER      DEFAULT NULL,
    p_user_name                    IN      VARCHAR2,
    p_first_name                   IN      VARCHAR2    DEFAULT NULL,
    p_last_name                    IN      VARCHAR2    DEFAULT NULL,
    p_description                  IN      VARCHAR2    DEFAULT NULL,
    p_email_address                IN      VARCHAR2    DEFAULT NULL,
    p_web_password                 IN      VARCHAR2,
    p_web_password_format          IN      VARCHAR2    DEFAULT 'CLEAR_TEXT',
    p_group_ids                    IN      VARCHAR2    DEFAULT NULL,
    p_developer_privs              IN      VARCHAR2    DEFAULT NULL,
    p_default_schema               IN      VARCHAR2    DEFAULT NULL,
    p_allow_access_to_schemas      IN      VARCHAR2    DEFAULT NULL,
```

```
p_account_expiry                IN      DATE        DEFAULT TRUNC(SYSDATE),
p_account_locked                IN      VARCHAR2    DEFAULT 'N',
p_failed_access_attempts        IN      NUMBER      DEFAULT 0,
p_change_password_on_first_use  IN      VARCHAR2    DEFAULT 'Y',
p_first_password_use_occurred   IN      VARCHAR2    DEFAULT 'N',
p_attribute_01                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_02                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_03                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_04                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_05                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_06                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_07                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_08                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_09                  IN      VARCHAR2    DEFAULT NULL,
p_attribute_10                  IN      VARCHAR2    DEFAULT NULL,
p_allow_app_building_yn         IN      VARCHAR2    DEFAULT NULL,
p_allow_sql_workshop_yn         IN      VARCHAR2    DEFAULT NULL,
p_allow_websheet_dev_yn         IN      VARCHAR2    DEFAULT NULL,
p_allow_team_development_yn     IN      VARCHAR2    DEFAULT NULL);
```

Parameters

Table 28-11 CREATE_USER Procedure Parameters

Parameter	Description
p_user_id	Numeric primary key of user account.
p_user_name	Alphanumeric name used for login.
p_first_name	Informational.
p_last_name	Informational.
p_description	Informational.
p_email_address	Email address.
p_web_password	Clear text password.
p_web_password_format	If the value your passing for the p_web_password parameter is in clear text format then use CLEAR_TEXT, otherwise use HEX_ENCODED_DIGEST_V2.
p_group_ids	Colon separated list of numeric group IDs.

Table 28-11 (Cont.) CREATE_USER Procedure Parameters

Parameter	Description
p_developer_privs	Colon separated list of developer privileges. If p_developer_privs is not null, the user is given access to Team Development. If p_developer_privs contains ADMIN, the user is given App Builder and SQL Workshop access. If p_developer_privs does not contain ADMIN but contains EDIT, the user is given App Builder Access. If p_developer_privs does not contain ADMIN but contains SQL, the user is given SQL Workshop access. The following are acceptable values for this parameter: **null** - To create an end user (a user who can only authenticate to developed applications). **CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL** - To create a user with developer privileges with access to App Builder and SQL Workshop. **ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL** - To create a user with full workspace administrator and developer privileges with access to App Builder, SQL Workshop and Team Development. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing.
p_allow_access_to_schemas	Colon separated list of schemas assigned to the user's workspace to which the user is restricted (leave null for all).
p_account_expiry	Date password was last updated, which defaults to today's date on creation.
p_account_locked	'Y' or 'N' indicating if account is locked or unlocked.
p_failed_access_attempts	Number of consecutive login failures that have occurred, defaults to 0 on creation.
p_change_password_on_first_use	'Y' or 'N' to indicate whether password must be changed on first use, defaults to 'Y' on creation.
p_first_password_use_occurred	'Y' or 'N' to indicate whether login has occurred since password change, defaults to 'N' on creation.

Table 28-11 (Cont.) CREATE_USER Procedure Parameters

Parameter	Description
p_attribute_01 ... p_attribute_10	Arbitrary text accessible with an API.
p_allow_app_building_yn	'Y' or 'N' to indicate whether access is allowed to App Builder.
p_allow_sql_workshop_yn	'Y' or 'N' to indicate whether access is allowed to SQL Workshop.
p_allow_websheet_dev_yn	'Y' or 'N' to indicate whether access is allowed to Websheet development.
p_allow_team_development_yn	'Y' or 'N' to indicate whether access is allowed to Team Development.

Example 1

The following simple example creates an 'End User' called 'NEWUSER1' with a password of 'secret99'. Note an 'End User' can only authenticate to developed applications.

```
BEGIN
    APEX_UTIL.CREATE_USER(
        p_user_name    => 'NEWUSER1',
        p_web_password => 'secret99');
END;
```

Example 2

The following example creates a 'Workspace Administrator' called 'NEWUSER2'. Where the user 'NEWUSER2':

- Has full workspace administration and developer privilege (`p_developer_privs` parameter set to '`ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL`').
- Has access to 2 schemas, both their browsing default '`MY_SCHEMA`' (`p_default_schema` parameter set to '`MY_SCHEMA`') and also '`MY_SCHEMA2`' (`p_allow_access_to_schemas` parameter set to '`MY_SCHEMA2`').
- Does not have to change their password when they first login (`p_change_password_on_first_use` parameter set to 'N').
- Has their phone number stored in the first additional attribute (`p_attribute_01` parameter set to '123 456 7890').

```
BEGIN
    APEX_UTIL.CREATE_USER(
        p_user_name                    => 'NEWUSER2',
        p_first_name                   => 'FRANK',
        p_last_name                    => 'SMITH',
        p_description                  => 'Description...',
```

```
        p_email_address                => 'frank@smith.com',
        p_web_password                 => 'password',
        p_developer_privs              =>
'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL',
        p_default_schema               => 'MY_SCHEMA',
        p_allow_access_to_schemas      => 'MY_SCHEMA2',
        p_change_password_on_first_use => 'N',
        p_attribute_01                 => '123 456 7890');
END;
```

See Also:

- "FETCH_USER Procedure Signature 3 (page 28-37)"
- "EDIT_USER Procedure (page 28-25)"
- "GET_GROUP_ID Function (page 28-52)"

28.14 CREATE_USER_GROUP Procedure

Assuming you are using Application Express authentication, this procedure creates a user group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.CREATE_USER_GROUP(
    p_id                    IN NUMBER default null,
    p_group_name            IN ARCHAR2,
    p_security_group_id     IN NUMBER default null,
    p_group_desc            IN VARCHAR2 default null,);
```

Parameter

Table 28-12 CREATE_USER_GROUP Parameters

Parameter	Description
p_id	Primary key of group.
p_group_name	Name of group.
p_security_group_id	Workspace ID.
p_group_desc	Descriptive text.

Example

The following example demonstrates how to use the CREATE_USER_GROUP procedure to create a new group called 'Managers' with a description of 'text'. Pass null for the p_id parameter to allow the database trigger to assign the new primary key value. Pass null for the p_security_group_id parameter to default to the current workspace ID.

```
BEGIN
    APEX_UTIL.CREATE_USER_GROUP (
```

```
        p_id               => null,         -- trigger assigns PK
        p_group_name       => 'Managers',
        p_security_group_id => null,        -- defaults to current workspace ID
        p_group_desc       => 'text');
END;
```

28.15 CURRENT_USER_IN_GROUP Function

This function returns a Boolean result based on whether the current user is a member of the specified group. You can use the group name or group ID to identify the group.

Syntax

```
APEX_UTIL.CURRENT_USER_IN_GROUP(
    p_group_name  IN VARCHAR2)
RETURN BOOLEAN;

APEX_UTIL.CURRENT_USER_IN_GROUP(
    p_group_id    IN NUMBER)
RETURN BOOLEAN;
```

Parameters

Table 28-13 CURRENT_USER_IN_GROUP Parameters

Parameter	Description
p_group_name	Identifies the name of an existing group in the workspace
p_group_id	Identifies the numeric ID of an existing group in the workspace

Example

The following example demonstrates how to use the CURRENT_USER_IN_GROUP function to check if the user currently authenticated belongs to the group 'Managers'.

```
DECLARE
    VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.CURRENT_USER_IN_GROUP(p_group_name=>'Managers');
END;
```

28.16 CUSTOM_CALENDAR Procedure

Use this procedure to change the existing calendar view to Custom Calendar.

Syntax

```
APEX_UTIL.CUSTOM_CALENDAR(
    p_date_type_field IN VARCHAR2);
```

Parameters

Table 28-14 CUSTOM_CALENDAR Parameters

Parameter	Description
p_date_type_field	Identifies the item name used to define the type of calendar to be displayed.

Example 1

The following example defines a custom calendar based on the hidden calendar type field. Assuming the Calendar is created in Page 9, the following example hides the column called P9_CALENDAR_TYPE.

```
APEX_UTIL.CUSTOM_CALENDAR(
    'P9_CALENDAR_TYPE');
```

28.17 DELETE_USER_GROUP Procedure Signature 1

Assuming you are using Application Express authentication, this procedure deletes a user group by providing the primary key of the group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.DELETE_USER_GROUP(
    p_group_id IN NUMBER);
```

Parameter

Table 28-15 DELETE_USER_GROUP Procedure Signature 1 Parameters

Parameter	Description
p_group_id	Primary key of group.

Example

The following example demonstrates how to use the DELETE_USER_GROUP procedure signature 1 to remove the user group called 'Managers', by providing the user group's primary key.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_GROUP_ID (
        p_group_name => 'Managers');
    APEX_UTIL.DELETE_USER_GROUP (
        p_group_id => VAL);
END;
```

28.18 DELETE_USER_GROUP Procedure Signature 2

Assuming you are using Application Express authentication, this procedure deletes a user group by providing the name of the group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.DELETE_USER_GROUP(
    p_group_name IN VARCHAR2);
```

Parameter

Table 28-16 DELETE_USER_GROUP Procedure Signature 2 Parameters

Parameter	Description
p_group_name	Name of group

Example

The following example demonstrates how to use the DELETE_USER_GROUP procedure signature 2 to remove the user group called 'Managers', by providing the name of the user group.

```
BEGIN
    APEX_UTIL.DELETE_USER_GROUP (
        p_group_name => 'Managers');
END;
```

28.19 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1

This procedure initiates the download of a print document using XML based report data (as a BLOB) and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_report_data         IN BLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 28-17 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_dispositio n	Specifies whether to download the print document or display inline ("attachment", "inline").
p_report_data	XML based report data.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".

Table 28-17 (Cont.) DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

See Also:

"Printing Report Regions" in *Oracle Application Express App Builder User's Guide*.

28.20 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2

This procedure initiates the download of a print document using pre-defined report query and RTF and XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 28-18 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_dispositio n	Specifies whether to download the print document or display inline ("attachment", "inline").
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's Shared Components).
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".

Table 28-18 (Cont.) DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example for Signature 2

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 2 (Pre-defined report query and RTF or XSL-FO based report layout.). In this example, the data for the report is taken from a Report Query called 'ReportQueryAndXSL' stored in the current application's Shared Components > Report Queries. The report layout is taken from a value stored in a page item (P1_XSL).

```
BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name           => 'mydocument',
        p_content_disposition => 'attachment',
        p_application_id      => :APP_ID,
        p_report_query_name   => 'ReportQueryAndXSL',
        p_report_layout       => :P1_XSL,
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');
END;
```

See Also:

"Printing Report Regions" in *Oracle Application Express App Builder User's Guide*.

28.21 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3

This procedure initiates the download of a print document using pre-defined report query and pre-defined report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout_name  IN VARCHAR2,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 28-19 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline").
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's Shared Components).
p_report_layout_name	Name of the report layout (stored under application's Shared Components).
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example for Signature 3

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 3 (Pre-defined report query and pre-defined report layout). In this example, the data for the report is taken from a Report Query called 'ReportQuery' stored in the current application's Shared Components > Report Queries. The report layout is taken from a Report Layout called 'ReportLayout' stored in the current application's Shared Components > Report Layouts. Note that if you want to provision dynamic layouts, instead of specifying 'ReportLayout' for the p_report_layout_name parameter, you could reference a page item that allowed the user to select one of multiple saved Report Layouts. This example also provides a way for the user to specify how they want to receive the document (as an attachment or inline), through passing the value of P1_CONTENT_DISP to the p_content_disposition parameter. P1_CONTENT_DISP is a page item of type 'Select List' with the following List of Values Definition:

```
STATIC2:In Browser;inline,Save / Open in separate Window;attachment

BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name           => 'myreport123',
        p_content_disposition => :P1_CONTENT_DISP,
        p_application_id      => :APP_ID,
        p_report_query_name   => 'ReportQuery',
        p_report_layout_name  => 'ReportLayout',
        p_report_layout_type  => 'rtf',
        p_document_format     => 'pdf');
END;
```

See Also:

"Printing Report Regions" in *Oracle Application Express App Builder User's Guide*.

28.22 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4

This procedure initiates the download of a print document using XML based report data (as a CLOB) and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_report_data         IN CLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 28-20 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline").
p_report_data	XML based report data, must be encoded in UTF-8.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example for Signature 4

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 4 (XML based report data (as a CLOB) and RTF or XSL-FO based report layout). In this example both the report data (XML) and report layout (XSL-FO) are taken from values stored in page items.

```
BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name           => 'mydocument',
        p_content_disposition => 'attachment',
        p_report_data         => :P1_XML,
        p_report_layout       => :P1_XSL,
```

```
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');
END;
```

See Also:

"Printing Report Regions" in *Oracle Application Express App Builder User's Guide*.

28.23 EDIT_USER Procedure

This procedure enables a user account record to be altered. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.EDIT_USER (
    p_user_id                      IN                  NUMBER,
    p_user_name                    IN                  VARCHAR2,
    p_first_name                   IN                  VARCHAR2    DEFAULT NULL,
    p_last_name                    IN                  VARCHAR2    DEFAULT NULL,
    p_web_password                 IN                  VARCHAR2    DEFAULT NULL,
    p_new_password                 IN                  VARCHAR2    DEFAULT NULL,
    p_email_address                IN                  VARCHAR2    DEFAULT NULL,
    p_start_date                   IN                  VARCHAR2    DEFAULT NULL,
    p_end_date                     IN                  VARCHAR2    DEFAULT NULL,
    p_employee_id                  IN                  VARCHAR2    DEFAULT NULL,
    p_allow_access_to_schemas      IN                  VARCHAR2    DEFAULT NULL,
    p_person_type                  IN                  VARCHAR2    DEFAULT NULL,
    p_default_schema               IN                  VARCHAR2    DEFAULT NULL,
    p_group_ids                    IN                  VARCHAR2    DEFAULT NULL,
    p_developer_roles              IN                  VARCHAR2    DEFAULT NULL,
    p_description                  IN                  VARCHAR2    DEFAULT NULL,
    p_account_expiry               IN                  DATE        DEFAULT NULL,
    p_account_locked               IN                  VARCHAR2    DEFAULT 'N',
    p_failed_access_attempts       IN                  NUMBER      DEFAULT 0,
    p_change_password_on_first_use IN                  VARCHAR2    DEFAULT 'Y',
    p_first_password_use_occurred  IN                  VARCHAR2    DEFAULT 'N');
```

Parameters

Table 28-21 EDIT_USER Parameters

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_user_name	Alphanumeric name used for login. **See Also**: "SET_USERNAME Procedure (page 28-110)"
p_first_name	Informational. **See Also**: "SET_FIRST_NAME Procedure (page 28-99)"
p_last_name	Informational. **See Also**: "SET_LAST_NAME Procedure (page 28-102)"

Table 28-21 (Cont.) EDIT_USER Parameters

Parameter	Description
p_web_password	Clear text password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.
p_new_password	Clear text new password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.
p_email_address	Informational. **See Also**: "SET_EMAIL Procedure (page 28-98)"
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which the user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing.
p_group_ids	Colon-separated list of numeric group IDs.
p_developer_roles	Colon-separated list of developer privileges. The following are acceptable values for this parameter: · **null** - To update the user to be an end user (a user who can only authenticate to developed applications). . **CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL** - To update the user to have developer privilege. . **ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL** - To update the user to have full workspace administrator and developer privilege. **Note**: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function (page 28-67)"
p_description	Informational.

Table 28-21 (Cont.) EDIT_USER Parameters

Parameter	Description
p_account_expiry	Date password was last updated. **See Also**: "EXPIRE_END_USER_ACCOUNT Procedure (page 28-29)", "EXPIRE_WORKSPACE_ACCOUNT Procedure (page 28-30)", "UNEXPIRE_END_USER_ACCOUNT Procedure (page 28-121)", "UNEXPIRE_WORKSPACE_ACCOUNT Procedure (page 28-122)"
p_account_locked	'Y' or 'N' indicating if account is locked or unlocked. **See Also**: "LOCK_ACCOUNT Procedure (page 28-79)", "UNLOCK_ACCOUNT Procedure (page 28-123)"
p_failed_access_attempts	Number of consecutive login failures that have occurred.
p_change_password_on_first_use	'Y' or 'N' to indicate whether password must be changed on first use. **See Also**: "CHANGE_PASSWORD_ON_FIRST_USE Function (page 28-9)"
p_first_password_use_occurred	'Y' or 'N' to indicate whether login has occurred since password change. **See Also**: "PASSWORD_FIRST_USE_OCCURRED Function (page 28-80)"

Example

The following example shows how to use the EDIT_USER procedure to update a user account. This example shows how you can use the EDIT_USER procedure to change the user 'FRANK' from a user with just developer privilege to a user with workspace administrator and developer privilege. Firstly, the FETCH_USER procedure is called to assign account details for the user 'FRANK' to local variables. These variables are then used in the call to EDIT_USER to preserve the details of the account, with the exception of the value for the p_developer_roles parameter, which is set to 'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL'.

```
DECLARE
    l_user_id                      NUMBER;
    l_workspace                    VARCHAR2(255);
    l_user_name                    VARCHAR2(100);
    l_first_name                   VARCHAR2(255);
    l_last_name                    VARCHAR2(255);
    l_web_password                 VARCHAR2(255);
    l_email_address                VARCHAR2(240);
    l_start_date                   DATE;
    l_end_date                     DATE;
    l_employee_id                  NUMBER(15,0);
    l_allow_access_to_schemas      VARCHAR2(4000);
    l_person_type                  VARCHAR2(1);
    l_default_schema               VARCHAR2(30);
    l_groups                       VARCHAR2(1000);
    l_developer_role               VARCHAR2(60);
```

```
    l_description                   VARCHAR2(240);
    l_account_expiry                DATE;
    l_account_locked                VARCHAR2(1);
    l_failed_access_attempts        NUMBER;
    l_change_password_on_first_use  VARCHAR2(1);
    l_first_password_use_occurred   VARCHAR2(1);
BEGIN
    l_user_id := APEX_UTIL.GET_USER_ID('FRANK');

APEX_UTIL.FETCH_USER(
    p_user_id                       => l_user_id,
    p_workspace                     => l_workspace,
    p_user_name                     => l_user_name,
    p_first_name                    => l_first_name,
    p_last_name                     => l_last_name,
    p_web_password                  => l_web_password,
    p_email_address                 => l_email_address,
    p_start_date                    => l_start_date,
    p_end_date                      => l_end_date,
    p_employee_id                   => l_employee_id,
    p_allow_access_to_schemas       => l_allow_access_to_schemas,
    p_person_type                   => l_person_type,
    p_default_schema                => l_default_schema,
    p_groups                        => l_groups,
    p_developer_role                => l_developer_role,
    p_description                   => l_description,
    p_account_expiry                => l_account_expiry,
    p_account_locked                => l_account_locked,
    p_failed_access_attempts        => l_failed_access_attempts,
    p_change_password_on_first_use  => l_change_password_on_first_use,
    p_first_password_use_occurred   => l_first_password_use_occurred);
APEX_UTIL.EDIT_USER (
    p_user_id                       => l_user_id,
    p_user_name                     => l_user_name,
    p_first_name                    => l_first_name,
    p_last_name                     => l_last_name,
    p_web_password                  => l_web_password,
    p_new_password                  => l_web_password,
    p_email_address                 => l_email_address,
    p_start_date                    => l_start_date,
    p_end_date                      => l_end_date,
    p_employee_id                   => l_employee_id,
    p_allow_access_to_schemas       => l_allow_access_to_schemas,
    p_person_type                   => l_person_type,
    p_default_schema                => l_default_schema,
    p_group_ids                     => l_groups,
    p_developer_roles               =>
'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL',
    p_description                   => l_description,
    p_account_expiry                => l_account_expiry,
    p_account_locked                => l_account_locked,
    p_failed_access_attempts        => l_failed_access_attempts,
    p_change_password_on_first_use  => l_change_password_on_first_use,
    p_first_password_use_occurred   => l_first_password_use_occurred);
END;
```

See Also:

"FETCH_USER Procedure Signature 3 (page 28-37)"

28.24 END_USER_ACCOUNT_DAYS_LEFT Function

Returns the number of days remaining before a end user account password expires. This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT (
    p_user_name IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 28-22 END_USER_ACCOUNT_DAYS_LEFT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the END_USER_ACCOUNT_DAYS_LEFT function. Use this function to determine the number of days remaining before an Application Express end user account in the current workspace expires.

```
DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        l_days_left := APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT(p_user_name =>
c1.user_name);
        htp.p('End User Account:'||c1.user_name||' expires in '||l_days_left||'
days.');
    END LOOP;
END;
```

See Also:

- "EXPIRE_END_USER_ACCOUNT Procedure (page 28-29)"
- "UNEXPIRE_END_USER_ACCOUNT Procedure (page 28-121)"

28.25 EXPIRE_END_USER_ACCOUNT Procedure

Expires the login account for use as a workspace end user. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.EXPIRE_END_USER_ACCOUNT (
    p_user_name IN VARCHAR2
    );
```

Parameters

Table 28-23 EXPIRE_END_USER_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the EXPIRE_END_USER_ACCOUNT procedure. Use this procedure to expire an Oracle Application Express account (workspace administrator, developer, or end user) in the current workspace. This action specifically expires the account for its use by end users to authenticate to developed applications, but it may also expire the account for its use by developers or administrators to log in to a workspace.

Note that this procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (select user_name from apex_users) LOOP
        APEX_UTIL.EXPIRE_END_USER_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now expired.');
    END LOOP;
END;
```

See Also:

"UNEXPIRE_END_USER_ACCOUNT Procedure (page 28-121)"

28.26 EXPIRE_WORKSPACE_ACCOUNT Procedure

Expires developer or workspace administrator login accounts. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.EXPIRE_WORKSPACE_ACCOUNT (
    p_user_name IN VARCHAR2
    );
```

Parameters

Table 28-24 EXPIRE_WORKSPACE_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the EXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to expire an Application Express account (workspace administrator, developer, or end user) in the current workspace. This action specifically expires the account for its use by developers or administrators to log in to

a workspace, but it may also expire the account for its use by end users to authenticate to developed applications.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) LOOP
        APEX_UTIL.EXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' is now expired.');
    END LOOP;
END;
```

See Also:

"UNEXPIRE_WORKSPACE_ACCOUNT Procedure (page 28-122)"

28.27 EXPORT_USERS Procedure

When called from a page, this procedure produces an export file of the current workspace definition, workspace users, and workspace groups. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.EXPORT_USERS(
    p_export_format IN VARCHAR2 DEFAULT 'UNIX');
```

Parameters

Table 28-25 EXPORT_USERS Parameters

Parameter	Description
p_export_format	Indicates how rows in the export file are formatted. Specify 'UNIX' to have the resulting file contain rows delimited by line feeds. Specify 'DOS' to have the resulting file contain rows delimited by carriage returns and line feeds.

Example

The following example shows how to use the EXPORT_USERS procedure. Call this procedure from a page to produce an export file containing the current workspace definition, list of workspace users and list of workspace groups. The file is formatted with rows delimited by line feeds.

```
BEGIN
    APEX_UTIL.EXPORT_USERS;
END;
```

28.28 FETCH_APP_ITEM Function

This function fetches session state for the current or specified application in the current or specified session.

Syntax

```
APEX_UTIL.FETCH_APP_ITEM(
    p_item    IN VARCHAR2,
```

```
    p_app     IN NUMBER DEFAULT NULL,
    p_session IN NUMBER DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 28-26 FETCH_APP_ITEM Parameters

Parameter	Description
p_item	The name of an application-level item (not a page item) whose current value is to be fetched.
p_app	The ID of the application that owns the item (leave null for the current application).
p_session	The session ID from which to obtain the value (leave null for the current session).

Example

The following example shows how to use the FETCH_APP_ITEM function to obtain the value of the application item 'F300_NAME' in application 300. As no value is passed for p_session, this defaults to the current session state value.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.FETCH_APP_ITEM(
        p_item => 'F300_NAME',
        p_app => 300);
END;
```

28.29 FETCH_USER Procedure Signature 1

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 1

```
APEX_UTIL.FETCH_USER (
    p_user_id                   IN                  NUMBER,
    p_workspace                 OUT                 VARCHAR2,
    p_user_name                 OUT                 VARCHAR2,
    p_first_name                OUT                 VARCHAR2,
    p_last_name                 OUT                 VARCHAR2,
    p_web_password              OUT                 VARCHAR2,
    p_email_address             OUT                 VARCHAR2,
    p_start_date                OUT                 VARCHAR2,
    p_end_date                  OUT                 VARCHAR2,
    p_employee_id               OUT                 VARCHAR2,
    p_allow_access_to_schemas   OUT                 VARCHAR2,
    p_person_type               OUT                 VARCHAR2,
    p_default_schema            OUT                 VARCHAR2,
    p_groups                    OUT                 VARCHAR2,
    p_developer_role            OUT                 VARCHAR2,
    p_description               OUT                 VARCHAR2 );
```

Parameters for Signature 1

Table 28-27 Fetch_User Parameters Signature 1

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_workspace	The name of the workspace.
p_user_name	Alphanumeric name used for login. **See Also**: "GET_USERNAME Function (page 28-68)"
p_first_name	Informational. **See Also**: "GET_FIRST_NAME Function (page 28-50)"
p_last_name	Informational. **See Also**: "GET_LAST_NAME Function (page 28-55)"
p_web_password	Obfuscated account password.
p_email_address	Email address. **See Also**: "GET_EMAIL Function (page 28-47)"
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing. **See Also**: "GET_DEFAULT_SCHEMA Function (page 28-46)"
p_groups	List of groups of which user is a member. **See Also**: "GET_GROUPS_USER_BELONGS_TO Function (page 28-52)" and "CURRENT_USER_IN_GROUP Function (page 28-18)"

Table 28-27 (Cont.) Fetch_User Parameters Signature 1

Parameter	Description
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function (page 28-67)"
p_description	Informational.

Example for Signature 1

The following example shows how to use the FETCH_USER procedure with Signature 1. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_workspace                 VARCHAR2(255);
    l_user_name                 VARCHAR2(100);
    l_first_name                VARCHAR2(255);
    l_last_name                 VARCHAR2(255);
    l_web_password              VARCHAR2(255);
    l_email_address             VARCHAR2(240);
    l_start_date                DATE;
    l_end_date                  DATE;
    l_employee_id               NUMBER(15,0);
    l_allow_access_to_schemas   VARCHAR2(4000);
    l_person_type               VARCHAR2(1);
    l_default_schema            VARCHAR2(30);
    l_groups                    VARCHAR2(1000);
    l_developer_role            VARCHAR2(60);
    l_description               VARCHAR2(240);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id                   => APEX_UTIL.GET_CURRENT_USER_ID,
        p_workspace                 => l_workspace,
        p_user_name                 => l_user_name,
        p_first_name                => l_first_name,
        p_last_name                 => l_last_name,
        p_web_password              => l_web_password,
        p_email_address             => l_email_address,
        p_start_date                => l_start_date,
        p_end_date                  => l_end_date,
```

```
        p_employee_id               => l_employee_id,
        p_allow_access_to_schemas   => l_allow_access_to_schemas,
        p_person_type               => l_person_type,
        p_default_schema            => l_default_schema,
        p_groups                    => l_groups,
        p_developer_role            => l_developer_role,
        p_description               => l_description);
END;
```

See Also:

- "EDIT_USER Procedure (page 28-25)"
- "GET_CURRENT_USER_ID Function (page 28-46)"

28.30 FETCH_USER Procedure Signature 2

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 2

```
APEX_UTIL.FETCH_USER (
    p_user_id                   IN              NUMBER,
    p_user_name                 OUT             VARCHAR2,
    p_first_name                OUT             VARCHAR2,
    p_last_name                 OUT             VARCHAR2,
    p_email_address             OUT             VARCHAR2,
    p_groups                    OUT             VARCHAR2,
    p_developer_role            OUT             VARCHAR2,
    p_description               OUT             VARCHAR2 );
```

Parameters for Signature 2

Table 28-28 Fetch_User Parameters Signature 2

Parameter	Description
p_user_id	Numeric primary key of the user account
p_user_name	Alphanumeric name used for login. **See Also**: "GET_USERNAME Function (page 28-68)"
p_first_name	Informational. **See Also:** "GET_FIRST_NAME Function (page 28-50)"
p_last_name	Informational. **See Also:** "GET_LAST_NAME Function (page 28-55)"
p_email_address	Email address. **See Also**: "GET_EMAIL Function (page 28-47)"

Table 28-28 (Cont.) Fetch_User Parameters Signature 2

Parameter	Description
p_groups	List of groups of which user is a member. **See Also:** "GET_GROUPS_USER_BELONGS_TO Function (page 28-52)" and "CURRENT_USER_IN_GROUP Function (page 28-18)"
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function (page 28-67)"
p_description	Informational

Example for Signature 2

The following example shows how to use the FETCH_USER procedure with Signature 2. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_user_name         VARCHAR2(100);
    l_first_name        VARCHAR2(255);
    l_last_name         VARCHAR2(255);
    l_email_address     VARCHAR2(240);
    l_groups            VARCHAR2(1000);
    l_developer_role    VARCHAR2(60);
    l_description       VARCHAR2(240);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id           => APEX_UTIL.GET_CURRENT_USER_ID,
        p_user_name         => l_user_name,
        p_first_name        => l_first_name,
        p_last_name         => l_last_name,
        p_email_address     => l_email_address,
        p_groups            => l_groups,
        p_developer_role    => l_developer_role,
        p_description       => l_description);
END;
```

See Also:

- "EDIT_USER Procedure (page 28-25)"
- "GET_CURRENT_USER_ID Function (page 28-46)"

28.31 FETCH_USER Procedure Signature 3

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 3

```
APEX_UTIL.FETCH_USER (
    p_user_id                       IN              NUMBER,
    p_workspace                     OUT             VARCHAR2,
    p_user_name                     OUT             VARCHAR2,
    p_first_name                    OUT             VARCHAR2,
    p_last_name                     OUT             VARCHAR2,
    p_web_password                  OUT             VARCHAR2,
    p_email_address                 OUT             VARCHAR2,
    p_start_date                    OUT             VARCHAR2,
    p_end_date                      OUT             VARCHAR2,
    p_employee_id                   OUT             VARCHAR2,
    p_allow_access_to_schemas       OUT             VARCHAR2,
    p_person_type                   OUT             VARCHAR2,
    p_default_schema                OUT             VARCHAR2,
    p_groups                        OUT             VARCHAR2,
    p_developer_role                OUT             VARCHAR2,
    p_description                   OUT             VARCHAR2,
    p_account_expiry                OUT             DATE,
    p_account_locked                OUT             VARCHAR2,
    p_failed_access_attempts        OUT             NUMBER,
    p_change_password_on_first_use  OUT             VARCHAR2,
    p_first_password_use_occurred   OUT             VARCHAR2 );
```

Parameters for Signature 3

Table 28-29 Fetch_User Parameters Signature 3

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_workspace	The name of the workspace.
p_user_name	Alphanumeric name used for login. **See Also**: "GET_USERNAME Function (page 28-68)"
p_first_name	Informational. **See Also**: "GET_FIRST_NAME Function (page 28-50)"

Table 28-29 (Cont.) Fetch_User Parameters Signature 3

Parameter	Description
p_last_name	Informational. **See Also**: "GET_LAST_NAME Function (page 28-55)"
p_web_password	Obfuscated account password.
p_email_address	Email address. **See Also**: "GET_EMAIL Function (page 28-47)"
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing. **See Also**: "GET_DEFAULT_SCHEMA Function (page 28-46)"
p_groups	List of groups of which user is a member. **See Also**: "GET_GROUPS_USER_BELONGS_TO Function (page 28-52)" and "CURRENT_USER_IN_GROUP Function (page 28-18)"
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: `null` - Indicates an end user (a user who can only authenticate to developed applications). `CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL` - Indicates a user with developer privilege. `ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL` - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the `CREATE_USER`, `EDIT_USER` and `FETCH_USER` APIs, although they all relate to the `DEVELOPER_ROLE` field stored in the named user account record. `CREATE_USER` uses `p_developer_privs`, `EDIT_USER` uses `p_developer_roles` and `FETCH_USER` uses `p_developer_role`. **See Also**: "GET_USER_ROLES Function (page 28-67)"
p_description	Informational.

Table 28-29 (Cont.) Fetch_User Parameters Signature 3

Parameter	Description
p_account_expiry	Date account password was last reset. **See Also**: "END_USER_ACCOUNT_DAYS_LEFT Function (page 28-29)" and "WORKSPACE_ACCOUNT_DAYS_LEFT Function (page 28-125)"
p_account_locked	Locked/Unlocked indicator Y or N. **See Also**: "GET_ACCOUNT_LOCKED_STATUS Function (page 28-41)"
p_failed_access_attempts	Counter for consecutive login failures.
p_change_password_on_first_use	Setting to force password change on first use Y or N.
p_first_password_use_occurred	Indicates whether login with password occurred Y or N.

Example for Signature 3

The following example shows how to use the FETCH_USER procedure with Signature 3. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_workspace                    VARCHAR2(255);
    l_user_name                    VARCHAR2(100);
    l_first_name                   VARCHAR2(255);
    l_last_name                    VARCHAR2(255);
    l_web_password                 VARCHAR2(255);
    l_email_address                VARCHAR2(240);
    l_start_date                   DATE;
    l_end_date                     DATE;
    l_employee_id                  NUMBER(15,0);
    l_allow_access_to_schemas      VARCHAR2(4000);
    l_person_type                  VARCHAR2(1);
    l_default_schema               VARCHAR2(30);
    l_groups                       VARCHAR2(1000);
    l_developer_role               VARCHAR2(60);
    l_description                  VARCHAR2(240);
    l_account_expiry               DATE;
    l_account_locked               VARCHAR2(1);
    l_failed_access_attempts       NUMBER;
    l_change_password_on_first_use VARCHAR2(1);
    l_first_password_use_occurred  VARCHAR2(1);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id                     => APEX_UTIL.GET_CURRENT_USER_ID,
        p_workspace                   => l_workspace,
        p_user_name                   => l_user_name,
        p_first_name                  => l_first_name,
        p_last_name                   => l_last_name,
        p_web_password                => l_web_password,
        p_email_address               => l_email_address,
```

```
        p_start_date                    => l_start_date,
        p_end_date                      => l_end_date,
        p_employee_id                   => l_employee_id,
        p_allow_access_to_schemas       => l_allow_access_to_schemas,
        p_person_type                   => l_person_type,
        p_default_schema                => l_default_schema,
        p_groups                        => l_groups,
        p_developer_role                => l_developer_role,
        p_description                   => l_description,
        p_account_expiry                => l_account_expiry,
        p_account_locked                => l_account_locked,
        p_failed_access_attempts        => l_failed_access_attempts,
        p_change_password_on_first_use  => l_change_password_on_first_use,
        p_first_password_use_occurred   => l_first_password_use_occurred);
END;
```

See Also:

- "EDIT_USER Procedure (page 28-25)"
- "GET_CURRENT_USER_ID Function (page 28-46)"

28.32 FIND_SECURITY_GROUP_ID Function

This function returns the numeric security group ID of the named workspace.

Syntax

```
APEX_UTIL.FIND_SECURITY_GROUP_ID(
    p_workspace    IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 28-30 FIND_SECURITY_GROUP_ID Parameters

Parameter	Description
p_workspace	The name of the workspace.

Example

The following example demonstrates how to use the `FIND_SECURITY_GROUP_ID` function to return the security group ID for the workspace called 'DEMOS'.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.FIND_SECURITY_GROUP_ID (p_workspace=>'DEMOS');
END;
```

28.33 FIND_WORKSPACE Function

This function returns the workspace name associated with a security group ID.

Syntax

```
APEX_UTIL.FIND_WORKSPACE(
    p_security_group_id    IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-31 FIND_WORKSPACE Parameters

Parameter	Description
p_security_group_id	The security group ID of a workspace.

Example

The following example demonstrates how to use the FIND_WORKSPACE function to return the workspace name for the workspace with a security group ID of 20.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.FIND_WORKSPACE (p_security_group_id =>'20');
END;
```

28.34 GET_ACCOUNT_LOCKED_STATUS Function

Returns TRUE if the account is locked and FALSE if the account is unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS (
     p_user_name IN VARCHAR2
     ) RETURN BOOLEAN;
```

Parameters

Table 28-32 GET_ACCOUNT_LOCKED_STATUS Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the GET_ACCOUNT_LOCKED_STATUS function. Use this function to check if an Application Express user account (workspace administrator, developer, or end user) in the current workspace is locked.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) loop
        IF APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS(p_user_name => c1.user_name) THEN
            HTP.P('User Account:'||c1.user_name||' is locked.');
        END IF;
    END LOOP;
END;
```

See Also:

- LOCK_ACCOUNT Procedure (page 28-79)
- UNLOCK_ACCOUNT Procedure (page 28-123)

28.35 GET_APPLICATION_STATUS Function

This function returns the current status of the application. Status values include AVAILABLE, AVAILABLE_W_EDIT_LINK, DEVELOPERS_ONLY, RESTRICTED_ACCESS, UNAVAILABLE, UNAVAILABLE_PLSQL, and UNAVAILABLE_URL.

Syntax

```
APEX_UTIL.GET_APPLICATION_STATUS(
    p_application_id IN NUMBER) RETURN VARCHAR2;
```

Parameters

Table 28-33 GET_APPLICATION_STATUS Parameters

Parameter	Description
p_application_id	The Application ID.

Example

```
declare
    l_status varchar2(100);
begin
    l_status := apex_util.get_application_status(
                    p_application_id => 117 );
    dbms_output.put_line( 'The current application status is: ' || l_status );
end;
```

See Also:

"Availability" in *Oracle Application Express App Builder User's Guide*

28.36 GET_ATTRIBUTE Function

This function returns the value of one of the attribute values (1 through 10) of a named user in the Application Express accounts table. Please note these are only accessible by using the APIs.

Syntax

```
APEX_UTIL.GET_ATTRIBUTE(
    p_username               IN VARCHAR2,
    p_attribute_number       IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 28-34 GET_ATTRIBUTE Parameters

Parameter	Description
p_username	User name in the account.
p_attribute_number	Number of attributes in the user record (1 through 10).

Example

The following example shows how to use the GET_ATTTIBUTE function to return the value for the 1st attribute for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.GET_ATTRIBUTE (
        p_username => 'FRANK',
        p_attribute_number => 1);
END;
```

See Also:

"SET_ATTRIBUTE Procedure (page 28-93)"

28.37 GET_AUTHENTICATION_RESULT Function

Use this function to retrieve the authentication result of the current session. Any authenticated user can call this function in a page request context.

Syntax

```
APEX_UTIL.GET_AUTHENTICATION_RESULT
RETURN NUMBER;
```

Parameters

None.

Example

The following example demonstrates how to use the post-authentication process of an application's authentication scheme to retrieve the authentication result code set during authentication.

```
APEX_UTIL.SET_SESSION_STATE('MY_AUTH_STATUS',
    'Authentication result:'||APEX_UTIL.GET_AUTHENTICATION_RESULT);
```

See Also:

- "SET_AUTHENTICATION_RESULT Procedure (page 28-94)"
- "SET_CUSTOM_AUTH_STATUS Procedure (page 28-97)"

28.38 GET_BLOB_FILE_SRC Function

As an alternative to using the built-in methods of providing a download link, you can use the APEX_UTIL.GET_BLOB_FILE_SRC function. One advantage of this approach, is the ability to more specifically format the display of the image (with height and width tags). Please note that this approach is only valid if called from a valid Oracle Application Express session. Also, this method requires that the parameters that describe the BLOB to be listed as the format of a valid item within the application. That item is then referenced by the function.

Syntax

```
APEX_UTIL.GET_BLOB_FILE_SRC (
    p_item_name           IN VARCHAR2 DEFAULT NULL,
    p_v1                  IN VARCHAR2 DEFAULT NULL,
    p_v2                  IN VARCHAR2 DEFAULT NULL,
    p_content_disposition IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 28-35 GET_BLOB_FILE_SRC Parameters

Parameter	Description
p_item_name	Name of valid application page ITEM that with type FILE that contains the source type of DB column.
p_v1	Value of primary key column 1.
p_v2	Value of primary key column 2.
p_content_disposition	Specify inline or attachment, all other values ignored.

Example

As a PLSQL Function Body:

```
RETURN '<img src="'||APEX_UTIL.GET_BLOB_FILE_SRC('P2_ATTACHMENT',:P2_EMPNO)||'" />';
```

As a Region Source of type SQL:

```
SELECT ID, NAME,CASE WHEN NVL(dbms_lob.getlength(document),0) = 0
    THEN NULL
    ELSE CASE WHEN attach_mimetype like 'image%'
    THEN '<img src="'||apex_util.get_blob_file_src('P4_DOCUMENT',id)||'" />'
    ELSE
    '<a href="'||apex_util.get_blob_file_src('P4_DOCUMENT',id)||'">Download</a>'
    end
    END new_img
    FROM TEST_WITH_BLOB
```

The previous example illustrates how to display the BLOB within the report, if it can be displayed, and provide a download link, if it cannot be displayed.

See Also:

- "About BLOB Support in Forms and Reports" in*Oracle Application Express App Builder User's Guide*
- "Running a Demonstration Application" in *Oracle Application Express App Builder User's Guide*

28.39 GET_BUILD_OPTION_STATUS Function Signature 1

Use this function to get the build option status of a specified application by providing the ID of the application build option.

Syntax

```
APEX_UTIL.GET_BUILD_OPTION_STATUS(
    p_application_id  IN NUMBER
    p_id              IN NUMBER;
```

Parameters

Table 28-36 GET_BUILD_OPTION_STATUS Function Signature 1 Paremeters

Parameters	Description
p_application_id	The ID of the application that owns the build option under shared components.
p_id	The ID of the build option in the application.

Example

The following code retrieves the current status of the specified build option that is identified by ID.

```
DECLARE
    l_status VARCHAR2(255);
BEGIN
    l_status := APEX_UTIL.GET_BUILD_OPTION_STATUS(
                    P_APPLICATION_ID => 101,
                    P_ID => 245935500311121039);
END;
/
```

28.40 GET_BUILD_OPTION_STATUS Function Signature 2

Use this function to get the build option status of a specified application by providing the name of the application build option.

Syntax

```
APEX_UTIL.GET_BUILD_OPTION_STATUS(
    p_application_id     IN NUMBER
    p_build_option_name IN VARCHAR2);
```

Parameters

Table 28-37 GET_BUILD_OPTION_STATUS Function Signature 2 Parameters

Parameters	Description
p_application_id	The ID of the application that owns the build option under shared components.
p_build_option_name	The name of the build option in the application.

Example

The following code retrieves the current status of the specified build option that is identified by name.

```
DECLARE
    l_status VARCHAR2(255);
BEGIN
    l_status := APEX_UTIL.GET_BUILD_OPTION_STATUS(
                    P_APPLICATION_ID => 101,
                    P_BUILD_OPTION_NAME => 'EXCLUDE_FROM_PRODUCTION');
END;
/
```

28.41 GET_CURRENT_USER_ID Function

This function returns the numeric user ID of the current user.

Syntax

```
APEX_UTIL.GET_CURRENT_USER_ID
RETURN NUMBER;
```

Parameters

None.

Example

This following example shows how to use the GET_CURRENT_USER_ID function. It returns the numeric user ID of the current user into a local variable.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_CURRENT_USER_ID;
END;
```

28.42 GET_DEFAULT_SCHEMA Function

This function returns the default schema name associated with the current user.

Syntax

```
APEX_UTIL.GET_DEFAULT_SCHEMA
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the GET_DEFAULT_SCHEMA function. It returns the default schema name associated with the current user into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_DEFAULT_SCHEMA;
END;
```

28.43 GET_EDITION Function

This function returns the edition for the current page view.

Syntax

```
APEX_UTIL.GET_EDITION
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the GET_EDITION function. It returns the edition name for the current page view into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_EDITION;
END;
```

28.44 GET_EMAIL Function

This function returns the email address associated with the named user.

Syntax

```
APEX_UTIL.GET_EMAIL(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 28-38 GET_EMAIL Parameters

Parameter	Description
p_username	The user name in the account.

Example

The following example shows how to use the GET_EMAIL function to return the email address of the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(240);
BEGIN
    VAL := APEX_UTIL.GET_EMAIL(p_username => 'FRANK');
END;
```

See Also:

"SET_EMAIL Procedure (page 28-98)"

28.45 GET_FEEDBACK_FOLLOW_UP Function

Use this function to retrieve any remaining follow up associated with a specific feedback.

Syntax

```
APEX_UTIL.GET_FEEDBACK_FOLLOW_UP (
    p_feedback_id    IN NUMBER,
    p_row            IN NUMBER DEFAULT 1,
    p_template       IN VARCHAR2 DEFAULT '<br />#CREATED_ON# (#CREATED_BY#)
#FOLLOW_UP#')
RETURN VARCHAR2;
```

Parameters

Table 28-39 GET_FEEDBACK_FOLLOW_UP Parameters

Parameter	Description
p_feedback_id	The unique identifier of the feedback item.
p_row	Identifies which follow-up to retrieve and is ordered by created_on_desc.
p_template	The template to use to return the follow up. Given the in the default template, the function can be used in a loop to return all the follow up to a feedback.

Example

The following example displays all the remaining follow-up for feedback with the ID of 123.

```
declare
   l_feedback_count  number;
begin
   select count(*)
     into l_feedback_count
     from apex_team_feedback_followup
    where feedback_id = 123;

   for i in 1..l_feedback_count loop
```

```
     htp.p(apex_util.get_feedback_follow_up (
              p_feedback_id => 123,
              p_row         => i,
              p_template    => '<br />#FOLLOW_UP# was created on #CREATED_ON# by
#CREATED_BY#') );
   end loop;
end;
/
```

28.46 GET_FILE Procedure

This procedure downloads files from the Oracle Application Express file repository. Please note if you are invoking this procedure during page processing, you must ensure that no page branch is invoked under the same condition, as it interferes with the file retrieval. This means that branches with any of the following conditions should not be set to fire:

- Branches with a 'When Button Pressed' attribute equal to the button that invokes the procedure.
- Branches with conditional logic defined that would succeed during page processing when the procedure is being invoked.
- As unconditional.

Syntax

```
APEX_UTIL.GET_FILE (
    p_file_id    IN   VARCHAR2,
    p_inline     IN   VARCHAR2 DEFAULT 'NO');
```

Parameters

Table 28-40 GET_FILE Parameters

Parameter	Description

p_file_id	ID in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace. The following example demonstrates how to use APEX_APPLICATION_FILES:
	
	`DECLARE`
	`  l_file_id NUMBER;`
	`BEGIN`
	`  SELECT id`
	`  INTO l_file_id`
	`  FROM APEX_APPLICATION_FILES`
	`  WHERE filename = 'myxml';`
	`  --`
	`  APEX_UTIL.GET_FILE(`
	`  p_file_id => l_file_id,`
	`  p_inline => 'YES');`
	`END;`

p_inline	Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment.

Example

The following example shows how to use the GET_FILE function to return the file identified by the ID 8675309. This is displayed inline in the browser.

```
BEGIN
    APEX_UTIL.GET_FILE(
        p_file_id   => '8675309',
        p_inline    => 'YES');
END;
```

See Also:

"GET_FILE_ID Function (page 28-50)"

28.47 GET_FILE_ID Function

This function obtains the primary key of a file in the Oracle Application Express file repository.

Syntax

```
APEX_UTIL.GET_FILE_ID (
    p_name    IN   VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 28-41 GET_FILE_ID Parameters

Parameter	Description
p_name	The NAME in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace.

Example

The following example shows how to use the GET_FILE_ID function to retrieve the database ID of the file with a filename of 'F125.sql'.

```
DECLARE
    l_name VARCHAR2(255);
    l_file_id NUMBER;
BEGIN
    SELECT name
        INTO l_name
        FROM APEX_APPLICATION_FILES
        WHERE filename = 'F125.sql';
        --
        l_file_id := APEX_UTIL.GET_FILE_ID(p_name => l_name);
END;
```

28.48 GET_FIRST_NAME Function

This function returns the FIRST_NAME field stored in the named user account record.

Syntax

```
APEX_UTIL.GET_FIRST_NAME
    p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-42 GET_FIRST_NAME Parameters

Parameter	Description
p_username	Identifies the user name in the account.

Example

The following example shows how to use the GET_FIRST_NAME function to return the FIRST_NAME of the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_FIRST_NAME(p_username => 'FRANK');
END;
```

See Also:

"SET_FIRST_NAME Procedure (page 28-99)"

28.49 GET_GLOBAL_NOTIFICATION Function

This function gets the global notification message which is the message displayed in page #GLOBAL_NOTIFICATION# substitution string.

Syntax

```
APEX_UTIL.GET_GLOBAL_NOTIFICATION(
   p_application_id IN NUMBER) RETURN VARCHAR2;
```

Parameters

Table 28-43 GET_GLOBAL_NOTIFICATION Parameters

Parameter	Description
p_application_id	The Application ID.

Example

```
declare
    l_global_notification varchar2(100);

begin
    l_global_notification := apex_util.get_global_notification(
                                  p_application_id => 117 );
    dbms_output.put_line( 'The current global notification is: ' ||
```

```
l_global_notification );
end;
```

See Also:

"Availability" in *Oracle Application Express App Builder User's Guide*

28.50 GET_GROUPS_USER_BELONGS_TO Function

This function returns a comma then a space separated list of group names to which the named user is a member.

Syntax

```
APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(
   p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-44 GET_GROUPS_USER_BELONGS_TO Parameters

Parameter	Description
p_username	Identifies the user name in the account.

Example

The following example shows how to use the GET_GROUPS_USER_BELONGS_TO to return the list of groups to which the user 'FRANK' is a member.

```
DECLARE
    VAL VARCHAR2(32765);
BEGIN
    VAL := APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'FRANK');
END;
```

See Also:

"EDIT_USER Procedure (page 28-25)"

28.51 GET_GROUP_ID Function

This function returns the numeric ID of a named group in the workspace.

Syntax

```
APEX_UTIL.GET_GROUP_ID(
    p_group_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-45 GET_GROUP_ID Parameters

Parameter	Description
p_group_name	Identifies the user name in the account.

Example

The following example shows how to use the GET_GROUP_ID function to return the ID for the group named 'Managers'.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_GROUP_ID(p_group_name => 'Managers');
END;
```

28.52 GET_GROUP_NAME Function

This function returns the name of a group identified by a numeric ID.

Syntax

```
APEX_UTIL.GET_GROUP_NAME(
    p_group_id IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 28-46 GET_GROUP_NAME Parameters

Parameter	Description
p_group_id	Identifies a numeric ID of a group in the workspace.

Example

The following example shows how to use the GET_GROUP_NAME function to return the name of the group with the ID 8922003.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;
```

28.53 GET_HASH Function

This function computes a hash value for all given values. Use this function to implement lost update detection for data records.

Syntax

```
APEX_UTIL.GET_HASH (
    p_values in apex_t_varchar2,
```

```
    p_salted in boolean default true )
    RETURN VARCHAR2;
```

Parameters

Table 28-47 GET_HASH Parameters

Parameter	Description
p_values	The input values.
p_salted	If true (the default), salt hash with internal session information.

Example

```
declare
       l_hash varchar2(4000);
   begin
       select apex_util.get_hash(apex_t_varchar2 (
                  empno, sal, comm ))
         into l_hash
         from emp
        where empno = :P1_EMPNO;

       if :P1_HASH <> l_hash then
           raise_application_error(-20001, 'Somebody already updated SAL/COMM');
       end if;

       update emp
          set sal = :P1_SAL,
              comm = :P1_COMM
        where empno = :P1_EMPNO;
   exception when no_data_found then
       raise_application_error(-20001, 'Employee not found');
   end;
```

28.54 GET_HIGH_CONTRAST_MODE_TOGGLE Function

This function returns a link to the current page that enables you to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches high contrast mode on.

Syntax

```
APEX_UTIL.GET_HIGH_CONTRAST_MODE_TOGGLE (
    p_on_message  IN VARCHAR2 DEFAULT NULL,
    p_off_message IN VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 28-48 GET_HIGH_CONTRAST_MODE_TOGGLE Prameters

Parameter	Description
p_on_message	Optional text used for the link to switch to high contrast mode, when you are in standard mode. If this parameter is not passed, the default 'Set High Contrast Mode On' text is returned in the link.
p_off_message	Optional text used for the link to switch to standard mode, when you are in high contrast mode. If this parameter is not passed, the default 'Set High Contrast Mode Off' text is returned in the link.

Example

When running in standard mode, this function returns a link with the text 'Set High Contrast Mode On'. When the link is clicked the current page is refreshed and high contrast mode is switched on. When running in high contrast mode, a link 'Set High Contrast Mode Off' is returned. When the link is clicked the current page is refreshed and switched back to standard mode.

```
BEGIN
    htp.p(apex_util.get_high_contrast_mode_toggle);
END;
```

Note:

There are also 2 translatable system messages that can be overridden at application level to change the default link text that is returned for this toggle. They include:

- APEX.SET_HIGH_CONTRAST_MODE_OFF - Default text = Set High Contrast Mode Off
- APEX.SET_HIGH_CONTRAST_MODE_ON - Default text = Set High Contrast Mode On

See Also:

"SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure (page 28-111)"

28.55 GET_LAST_NAME Function

This function returns the LAST_NAME field stored in the named user account record.

Syntax

```
APEX_UTIL.GET_LAST_NAME(
    p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-49 GET_LAST_NAME Parameters

Parameter	Description
p_username	The user name in the user account record.

Example

The following example shows how to use the function to return the LAST_NAME for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_LAST_NAME(p_username => 'FRANK');
END;
```

See Also:

"SET_LAST_NAME Procedure (page 28-102)"

28.56 GET_NUMERIC_SESSION_STATE Function

This function returns a numeric value for a numeric item. You can use this function in Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function NV, in place of APEX_UTIL.GET_NUMERIC_SESSION_STATE.

Syntax

```
APEX_UTIL.GET_NUMERIC_SESSION_STATE (
    p_item     IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 28-50 GET_NUMERIC_SESSION_STATE Parameters

Parameter	Description
p_item	The case insensitive name of the item for which you want to have the session state fetched.

Example

The following example shows how to use the function to return the numeric value stored in session state for the item 'my_item'.

```
DECLARE
    l_item_value    NUMBER;
BEGIN
    l_item_value := APEX_UTIL.GET_NUMERIC_SESSION_STATE('my_item');
END;
```

See Also:

- "GET_SESSION_STATE Function (page 28-62)"
- "SET_SESSION_STATE Procedure (page 28-108)"

28.57 GET_PREFERENCE Function

This function retrieves the value of a previously saved preference for a given user.

Syntax

```
APEX_UTIL.GET_PREFERENCE (
    p_preference  IN    VARCHAR2 DEFAULT NULL,
    p_user        IN    VARCHAR2 DEFAULT V('USER'))
RETURN VARCHAR2;
```

Parameters

Table 28-51 GET_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference to retrieve the value.
p_user	User for whom the preference is being retrieved.

Example

The following example shows how to use the GET_PREFERENCE function to return the value for the currently authenticated user's preference named default_view.

```
DECLARE
    l_default_view    VARCHAR2(255);
BEGIN
    l_default_view := APEX_UTIL.GET_PREFERENCE(
        p_preference => 'default_view',
        p_user       => :APP_USER);
END;
```

See Also:

- "SET_PREFERENCE Procedure (page 28-102)"
- "REMOVE_PREFERENCE Procedure (page 28-85)"
- "Managing User Preferences" in *Oracle Application Express Administration Guide*

28.58 GET_PRINT_DOCUMENT Function Signature 1

This function returns a document as BLOB using XML based report data and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_report_data         IN BLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default NULL)
RETURN BLOB;
```

Parameters

Table 28-52 GET_PRINT_DOCUMENT Signature 1 Parameters

Parameter	Description
p_report_data	XML based report data.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4 (page 28-60)".

28.59 GET_PRINT_DOCUMENT Function Signature 2

This function returns a document as BLOB using pre-defined report query and pre-defined report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout_name  IN VARCHAR2 default null,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null)
RETURN BLOB;
```

Parameters

Table 28-53 GET_PRINT_DOCUMENT Signature 2 Parameters

Parameter	Description
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's shared components).

Table 28-53 (Cont.) GET_PRINT_DOCUMENT Signature 2 Parameters

Parameter	Description
p_report_layout_name	Name of the report layout (stored under application's Shared Components).
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4 (page 28-60)".

28.60 GET_PRINT_DOCUMENT Function Signature 3

This function returns a document as BLOB using a pre-defined report query and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null)
RETURN BLOB;
```

Parameters

Table 28-54 GET_PRINT_DOCUMENT Signature 3 Parameters

Parameter	Description
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's shared components).
p_report_layout	Defines the report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4 (page 28-60)".

28.61 GET_PRINT_DOCUMENT Function Signature 4

This function returns a document as BLOB using XML based report data and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_report_data         IN CLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default NULL)
RETURN BLOB;
```

Parameters

Table 28-55 GET_PRINT_DOCUMENT Signature 4 Parameters

Parameter	Description
p_report_data	XML based report data, must be encoded in UTF-8.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example for Signature 4

The following example shows how to use the GET_PRINT_DOCUMENT using Signature 4 (Document returns as a BLOB using XML based report data and RTF or XSL-FO based report layout). In this example, GET_PRINT_DOCUMENT is used with APEX_MAIL.SEND and APEX_MAIL.ADD_ATTACHMENT to send an email with an attachment of the file returned by GET_PRINT_DOCUMENT. Both the report data and layout are taken from values stored in page items (P1_XML and P1_XSL).

```
DECLARE
    l_id number;
    l_document BLOB;
BEGIN
    l_document := APEX_UTIL.GET_PRINT_DOCUMENT (
        p_report_data         => :P1_XML,
        p_report_layout       => :P1_XSL,
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');

   l_id := APEX_MAIL.SEND(
       p_to        => :P35_MAIL_TO,
       p_from      => 'noreplies@oracle.com',
       p_subj      => 'sending PDF by using print API',
       p_body      => 'Please review the attachment.',
       p_body_html => 'Please review the attachment');
```

```
   APEX_MAIL.ADD_ATTACHMENT (
       p_mail_id    => l_id,
       p_attachment => l_document,
       p_filename   => 'mydocument.pdf',
       p_mime_type  => 'application/pdf');
END;
```

28.62 GET_SCREEN_READER_MODE_TOGGLE Function

This function returns a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches screen reader mode on.

Syntax

```
APEX_UTIL.GET_SCREEN_READER_MODE_TOGGLE (
    p_on_message  IN VARCHAR2 DEFAULT NULL,
    p_off_message IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 28-56 GET_SCREEN_READER_MODE_TOGGLE Parameters

Parameter	Description
p_on_message	Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text is returned in the link.
p_off_message	Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text is returned in the link.

Example

When running in standard mode, this function returns a link with the text 'Set Screen Reader Mode On'. When the link is clicked the current page is refreshed and screen reader mode is switched on. When running in screen reader mode, a link 'Set Screen Reader Mode Off' is returned. When the link is clicked the current page is refreshed and switched back to standard mode.

```
BEGIN
    htp.p(apex_util.get_screen_reader_mode_toggle);
END;
```

See Also:

"SHOW_SCREEN_READER_MODE_TOGGLE Procedure (page 28-112)"

28.63 GET_SESSION_LANG Function

This function returns the language setting for the current user in the current Application Express session.

Syntax

```
APEX_UTIL.GET_SESSION_LANG
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the GET_SESSION_LANG function. It returns the session language for the current user in the current Application Express session into a local variable.

```
DECLARE
    VAL VARCHAR2(5);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_LANG;
END;
```

28.64 GET_SESSION_STATE Function

This function returns the value for an item. You can use this function in your Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function `V`, in place of `APEX_UTIL.GET_SESSION_STATE`.

Syntax

```
APEX_UTIL.GET_SESSION_STATE (
    p_item    IN   VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-57 GET_SESSION_STATE Parameters

Parameter	Description
`p_item`	The case insensitive name of the item for which you want to have the session state fetched.

Example

The following example shows how to use the `GET_SESSION_STATE` function to return the value stored in session state for the item 'my_item'.

```
DECLARE
    l_item_value  VARCHAR2(255);
BEGIN
    l_item_value := APEX_UTIL.GET_SESSION_STATE('my_item');
END;
```

See Also:

- "GET_NUMERIC_SESSION_STATE Function (page 28-56)"
- "SET_SESSION_STATE Procedure (page 28-108)"

28.65 GET_SESSION_TERRITORY Function

This function returns the territory setting for the current user in the current Application Express session.

Syntax

```
APEX_UTIL.GET_SESSION_TERRITORY
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the `GET_SESSION_TERRITORY` function. It returns the session territory setting for the current user in the current Application Express session into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_TERRITORY;
END;
```

28.66 GET_SESSION_TIME_ZONE Function

This function returns the time zone for the current user in the current Application Express session. This value is null if the time zone is not explicitly set by using `APEX_UTIL.SET_SESSION_TIME_ZONE` or if an application's automatic time zone attribute is enabled.

Syntax

```
APEX_UTIL.GET_SESSION_TIME_ZONE
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the `GET_SESSION_TIME_ZONE` function. It returns the session time zone for the current user in the current Application Express session into a local variable.

```
BEGIN
    VAL := APEX_UTIL.GET_SESSION_TIME_ZONE;
END;
```

28.67 GET_SINCE Function

This function returns the relative date in words (for example, 2 days from now, 30 minutes ago). It also accepts a second optional `p_short` parameter and returns "in 2d", "30m". This function is equivalent to using the `SINCE` and `SINCE_SHORT` format mask available within Oracle Application Express and is useful within SQL queries or PL/SQL routines.

Syntax

```
APEX_UTIL.GET_SINCE (
   p_date  date )
   p_short in [ boolean default false | varchar2 default 'N' ] )
RETURN VARCHAR2;
```

Parameters

Table 28-58 GET_SINCE Parameters

Parameter	Description
`p_date`	The date you want formatted.
`p_short`	Boolean or 'Y' / 'N' to indicate whether to return a short version of relative date.

Example

```
select application_id, application_name,apex_util.get_since(last_updated_on)
last_update
  from apex_applications
 order by application_id
```

Syntax

```
APEX_UTIL.GET_SINCE (
   p_value in [ timestamp | timestamp with time zone | timestamp with local time
zone ],
   p_short in [ boolean default false | varchar2 default 'N' ] )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
`p_value`	The TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE you want formatted.
`p_short`	Boolean or 'Y' / 'N' to indicate whether to return a short version of relative date.

Example

This returns the `LAST_UPDATE` column with the normal formatting.

```
select application_id, application_name, apex_util.get_since( last_updated_on )
last_update
  from apex_applications
 order by application_id;
```

This returns the LAST_UPDATE column with the short formatting.

```
select application_id, application_name, apex_util.get_since( last_updated_on,
p_short => 'Y' ) last_update
  from apex_applications
 order by application_id
```

28.68 GET_SUPPORTING_OBJECT_SCRIPT Function

This function gets supporting object scripts defined in an application.

Note:

The workspace ID must be set before the call.

Syntax

```
APEX_UTIL.GET_SUPPORTING_OBJECT_SCRIPT (
    p_application_id  in number,
    p_script_type     in varchar2 ) return clob;
```

Parameters

Table 28-59 GET_SUPPORTING_OBJECT_SCRIPT Function

Parameter	Description
p_application_id	The application ID to get supporting objects from.
p_script_type	The supporting objects script type. Valid values are apex_util.c_install_script, apex_util.c_upgrade_script, apex_util.c_deinstall_script.

Example

The following example shows how to set workspace ID for workspace FRED, then get supporting objects from application ID 100.

```
declare
    l_install_script   clob;
    l_upgrade_script   clob;
    l_deinstall_script clob;
begin
    apex_util.set_workspace( p_workspace => 'FRED');

    l_install_script := apex_util.get_supporting_object_script( p_application_id =>
100,
 p_script_type => apex_util.c_install_script );
    l_upgrade_script := apex_util.get_supporting_object_script( p_application_id =>
```

```
100,
 p_script_type => apex_util.c_upgrade_script );
    l_deinstall_script := apex_util.get_supporting_object_script( p_application_id
=> 100,
 p_script_type => apex_util.c_deinstall_script );
end;
```

28.69 GET_SUPPORTING_OBJECT_SCRIPT Procedure

This procedure gets supporting object scripts and outputs to `sys.dbms_output` buffer or download as a file.

Note:

The workspace ID must be set before the call.

Syntax

```
APEX_UTIL.GET_SUPPORTING_OBJECT_SCRIPT(
    p_application_id  in number,
    p_script_type     in varchar2,
    p_output_type     in varchar2 default c_output_as_dbms_output );
```

Parameters

Table 28-60 GET_SUPPORTING_OBJECT_SCRIPT Procedure

Parameter	Description
`p_application_id`	The application ID to get supporting objects from.
`p_script_type`	The supporting objects script type. Valid values are `apex_util.c_install_script`, `apex_util.c_upgrade_script`, `apex_util.c_deinstall_script`.
`p_output_type`	The script can be output to `sys.dbms_output` buffer or download as a file. Valid values are `apex_util.c_output_as_dbms_output`, `apex_util.c_output_as_file`. The default is `c_output_as_dbms_output`.

Examples

The following example shows how to set workspace ID for workspace `FRED`, then get install script from application `ID 100` and output to the command-line buffer.

```
set serveroutput on;
begin
    apex_util.set_workspace( p_workspace => 'FRED');
    apex_util.get_supporting_object_script(
        p_application_id => 100,
```

```
        p_script_type    => apex_util.c_install_script );
end;
```

The following example shows how to download upgrade script file from application `ID 100` in the browser. Useful if the script needs to be downloaded using an application process.

```
begin
    apex_util.set_workspace( p_workspace => 'FRED');
    apex_util.get_supporting_object_script(
        p_application_id => 100,
        p_script_type    => apex_util.c_upgrade_script,
        p_output_type    => apex_util.c_output_as_file );
end;
```

28.70 GET_USER_ID Function

This function returns the numeric ID of a named user in the workspace.

Syntax

```
APEX_UTIL.GET_USER_ID(
    p_username   IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 28-61 GET_USER_ID Parameters

Parameter	Description
`p_username`	Identifies the name of a user in the workspace.

Example

The following example shows how to use the `GET_USER_ID` function to return the ID for the user named 'FRANK'.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_USER_ID(p_username => 'FRANK');
END;
```

28.71 GET_USER_ROLES Function

This function returns the `DEVELOPER_ROLE` field stored in the named user account record. Please note that currently this parameter is named inconsistently between the `CREATE_USER`, `EDIT_USER` and `FETCH_USER` APIs, although they all relate to the `DEVELOPER_ROLE` field. `CREATE_USER` uses `p_developer_privs`, `EDIT_USER` uses `p_developer_roles` and `FETCH_USER` uses `p_developer_role`.

Syntax

```
APEX_UTIL.GET_USER_ROLES(
   p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-62 GET_USER_ROLES Parameters

Parameter	Description
p_username	Identifies a user name in the account.

Example

The following example shows how to use the GET_USER_ROLES function to return colon separated list of roles stored in the DEVELOPER_ROLE field for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.GET_USER_ROLES(p_username=>'FRANK');
END;
```

28.72 GET_USERNAME Function

This function returns the user name of a user account identified by a numeric ID.

Syntax

```
APEX_UTIL.GET_USERNAME(
    p_userid IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 28-63 GET_USERNAME Parameters

Parameter	Description
p_userid	Identifies the numeric ID of a user account in the workspace.

Example

The following example shows how to use the GET_USERNAME function to return the user name for the user with an ID of 228922003.

```
DECLARE
    VAL VARCHAR2(100);
BEGIN
    VAL := APEX_UTIL.GET_USERNAME(p_userid => 228922003);
END;
```

See Also:

"SET_USERNAME Procedure (page 28-110)"

28.73 HOST_URL Function

This function returns the URL to the Application Express instance, depending on the option passed.

Syntax

```
APEX_UTIL.HOST_URL (
    p_option IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 28-64 HOST_URL Parameters

Parameter	Description

p_option	Specifies the parts of the URL to include.
	Possible values for p_option include:
	• NULL - Return URL up to port number. For example:
	`http://myserver.com:7778`
	• SCRIPT - Return URL to include script name. For example:
	`https://myserver.com:7778/pls/apex/`
	• IMGPRE - Return URL to include image prefix. For example:
	`https://myserver.com:7778/i/`

Example

The following example demonstrates how to use the HOST_URL function to return the URL, including the script name, to the current Application Express instance.

```
declare
    l_host_url      varchar2(4000);
    l_url           varchar2(4000);
    l_application   varchar2(30) := 'f?p=100:1';
    l_email_body    varchar2(32000);
begin
    l_host_url := apex_util.host_url('SCRIPT');
    l_url := l_host_url||l_application;
    l_email_body := 'The URL to the application is: '||l_url;
end;
```

28.74 HTML_PCT_GRAPH_MASK Function

Use this function to scale a graph. This function can also be used by classic and interactive reports with format mask of GRAPH. This generates a `<div>` tag with inline styles.

Syntax

```
APEX_UTIL.HTML_PCT_GRAPH_MASK (
    p_number         IN NUMBER    DEFAULT NULL,
    p_size           IN NUMBER    DEFAULT 100,
```

```
    p_background     IN VARCHAR2  DEFAULT NULL,
    p_bar_background IN VARCHAR2  DEFAULT NULL,
    p_format         IN VARCHAR2  DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 28-65 HTML_PCT_GRAPH_MASK Parameters

Parameter	Description
p_number	Number between 0 and 100.
p_size	Width of graph in pixels.
p_background	Six character hexadecimal background color of chart bar (not bar color).
p_bar_background	Six character hexadecimal background color of chart bar (bar color).
p_format	If this parameter is supplied, `p_size`, `p_background` and `p_bar_background` are ignored. This parameter uses the following format: PCT_GRAPH:<BACKGROUND>:<FOREGROUND>:<CHART_WIDTH> position 1: PCT_GRAPH format mask indicator position 2: Background color in hexadecimal, 6 characters (optional) position 3: Foreground "bar" color in hexadecimal, 6 characters (optional) position 4: Chart width in pixels. Numeric and defaults to 100. `p_number` is automatically scaled so that 50 is half of `chart_width` (optional).

Example

The following is an SQL example.

```
select apex_util.html_pct_graph_mask(33) from dual
```

The following is a report numeric column format mask example.

```
PCT_GRAPH:777777:111111:200
```

28.75 INCREMENT_CALENDAR Procedure

Use this procedure to navigate to the next set of days in the calendar. Depending on what the calendar view is, this procedure navigates to the next month, week or day. If it is a Custom Calendar the total number of days between the start date and end date are navigated.

Syntax

```
APEX_UTIL.INCREMENT_CALENDAR;
```

Parameter

None.

Example

In this example, if you create a button called NEXT in the Calendar page and create a process that fires when the create button is clicked the following code navigates the calendar.

```
APEX_UTIL.INCREMENT_CALENDAR
```

28.76 IR_CLEAR Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure clears report settings.

Note:

This procedure should be used only in a page submit process.

Syntax

```
APEX_UTIL.IR_CLEAR(
    p_page_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-66 IR_CLEAR Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_report_alias	Identifies the saved report alias within the current application page. To clear a Primary report, p_report_alias must be 'PRIMARY' or leave as NULL. To clear a saved report, p_report_alias must be the name of the saved report. For example, to clear report '1234', p_report_alias must be '1234'.

Example

The following example shows how to use the IR_CLEAR procedure to clear Interactive report settings with alias of '8101021' in page 1 of the current application.

```
BEGIN
    APEX_UTIL.IR_CLEAR(
        p_page_id      => 1,
        p_report_alias => '8101021'
        );
END;
```

28.77 IR_DELETE_REPORT Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure deletes saved Interactive reports. It deletes all saved reports except the Primary Default report.

Syntax

```
APEX_UTIL.IR_DELETE_REPORT(
    p_report_id IN NUMBER);
```

Parameters

Table 28-67 IR_DELETE_REPORT Parameters

Parameter	Description
p_report_id	Report ID to delete within the current Application Express application.

Example

The following example shows how to use the IR_DELETE_REPORT procedure to delete the saved Interactive report with ID of '880629800374638220' in the current application.

```
BEGIN
    APEX_UTIL.IR_DELETE_REPORT(
        p_report_id => '880629800374638220');
END;
```

28.78 IR_DELETE_SUBSCRIPTION Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure deletes Interactive subscriptions.

Syntax

```
APEX_UTIL.IR_DELETE_SUBSCRIPTION(
    p_subscription_id IN NUMBER);
```

Parameters

Table 28-68 IR_DELETE_SUBSCRIPTION Parameters

Parameter	Description
p_subscription_id	Subscription ID to delete within the current workspace.

Example

The following example shows how to use the IR_DELETE_SUBSCRIPTION procedure to delete the subscription with ID of ' 880629800374638220 ' in the current workspace.

```
BEGIN
    APEX_UTIL.IR_DELETE_SUBSCRIPTION(
        p_subscription_id => '880629800374638220');
END;
```

28.79 IR_FILTER Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure creates a filter on an interactive report.

Note:

This procedure should be used only in a page submit process.

Syntax

```
APEX_UTIL.IR_FILTER(
    p_page_id       IN NUMBER,
    p_report_column IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_filter_value  IN VARCHAR2,
    p_report_alias  IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-69 IR_FILTER Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_report_column	Name of the report SQL column, or column alias, to be filtered.

Table 28-69 (Cont.) IR_FILTER Parameters

Parameter	Description
p_operator_abbr	Filter type. Valid values are as follows: • EQ = Equals • NEQ = Not Equals • LT = Less than • LTE = Less then or equal to • GT = Greater Than • GTE = Greater than or equal to • LIKE = SQL Like operator • N = Null • NN = Not Null • C = Contains • NC = Not Contains • IN = SQL In Operator • NIN = SQL Not In Operator
p_filter_value	Filter value. This value is not used for 'N' and 'NN'.
p_report_alias	Identifies the saved report alias within the current application page. To create a filter on a Primary report, p_report_alias must be 'PRIMARY' or leave as NULL. To create a filter on a saved report, p_report_alias must be the name of the saved report. For example, to create a filter on report '1234', p_report_alias must be '1234'.

Example

The following example shows how to use the IR_FILTER procedure to filter Interactive report with alias of '8101021' in page 1 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_UTIL.IR_FILTER (
        p_page_id        => 1,
        p_report_column  => 'DEPTNO',
        p_operator_abbr  => 'EQ',
        p_filter_value   => '30'
        p_report_alias   => '8101021'
        );
END;
```

28.80 IR_RESET Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure resets report settings back to the default report settings. Resetting a report removes any customizations you have made.

Note:

This procedure should be used only in a page submit process.

Syntax

```
APEX_UTIL.IR_RESET(
    p_page_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-70 IR_RESET Parameters

Parameter	Description
p_page_id	Page of the current Application Express application that contains an interactive report.
p_report_alias	Identifies the saved report alias within the current application page. To reset a Primary report, p_report_alias must be 'PRIMARY' or leave as NULL. To reset a saved report, p_report_alias must be the name of the saved report. For example, to reset report '1234', p_report_alias must be '1234'.

Example

The following example shows how to use the IR_RESET procedure to reset Interactive report settings with alias of '8101021' in page 1 of the current application.

```
BEGIN
    APEX_UTIL.IR_RESET(
        p_page_id      => 1,
        p_report_alias => '8101021'
        );
END;
```

28.81 IS_HIGH_CONTRAST_SESSION Function

This function returns a boolean TRUE if the session is in high contrast mode and returns a boolean FALSE if not in high contrast mode.

Syntax

```
APEX_UTIL.IS_HIGH_CONTRAST_SESSION
RETURN BOOLEAN;
```

Parameters

None.

Example

In this example, if the current session is running in high contrast mode, a high contrast specific CSS file 'my_app_hc.css' is added to the HTML output of the page.

```
BEGIN
    IF apex_util.is_high_contrast_session THEN
        apex_css.add_file (
            p_name => 'my_app_hc');
    END IF;
END;
```

28.82 IS_HIGH_CONTRAST_SESSION_YN Function

This function returns `Y` if the session is in high contrast mode and `N` if not in high contrast mode.

Syntax

```
APEX_UTIL.IS_HIGH_CONTRAST_SESSION_YN
RETURN VARCHAR2;
```

Parameters

None.

Example

In this example, if the current session is running in high contrast mode, a high contrast specific CSS file, `my_app_hc.css`, is added to the HTML output of the page.

```
BEGIN
    IF apex_util.is_high_contrast_session_yn = 'Y' THEN
        apex_css.add_file (
            p_name => 'my_app_hc');
    END IF;
END;
```

28.83 IS_LOGIN_PASSWORD_VALID Function

This function returns a Boolean result based on the validity of the password for a named user account in the current workspace. This function returns TRUE if the password matches and it returns FALSE if the password does not match.

Syntax

```
APEX_UTIL.IS_LOGIN_PASSWORD_VALID(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 28-71 IS_LOGIN_PASSWORD_VALID Parameters

Parameter	Description
`p_username`	User name in account.
`p_password`	Password to be compared with password stored in the account.

Example

The following example shows how to use the IS_LOGIN_PASSWORD_VALID function to check if the user 'FRANK' has the password 'tiger'. TRUE is returned if this is a valid password for 'FRANK', FALSE is returned if not.

```
DECLARE
    VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.IS_LOGIN_PASSWORD_VALID (
        p_username=>'FRANK',
        p_password=>'tiger');
END;
```

28.84 IS_SCREEN_READER_SESSION Function

This function returns a boolean TRUE if the session is in screen reader mode and returns a boolean FALSE if not in screen reader mode.

Syntax

```
APEX_UTIL.IS_SCREEN_READER_SESSION
RETURN BOOLEAN;
```

Parameters

None

Example

```
BEGIN
    IF apex_util.is_screen_reader_session then
        htp.p('Screen Reader Mode');
    END IF;
END;
```

28.85 IS_SCREEN_READER_SESSION_YN Function

This function returns 'Y' if the session is in screen reader mode and 'N' if not in screen reader mode.

Syntax

```
APEX_UTIL.IS_SCREEN_READER_SESSION_YN
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
    IF apex_util.is_screen_reader_session_yn = 'Y' then
        htp.p('Screen Reader Mode');
    END IF;
END;
```

28.86 IS_USERNAME_UNIQUE Function

This function returns a Boolean result based on whether the named user account is unique in the workspace.

Syntax

```
APEX_UTIL.IS_USERNAME_UNIQUE(
    p_username IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 28-72 IS_USERNAME_UNIQUE Parameters

Parameter	Description
p_username	Identifies the user name to be tested.

Example

The following example shows how to use the IS_USERNAME_UNIQUE function. If the user 'FRANK' already exists in the current workspace, FALSE is returned, otherwise TRUE is returned.

```
DECLARE
    VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.IS_USERNAME_UNIQUE(
        p_username=>'FRANK');
END;
```

28.87 KEYVAL_NUM Function

This function gets the value of the package variable (apex_utilities.g_val_num) set by APEX_UTIL.SAVEKEY_NUM.

Syntax

```
APEX_UTIL.KEYVAL_NUM
RETURN NUMBER;
```

Parameters

None

Example

The following example shows how to use the KEYVAL_NUM function to return the current value of the package variable apex_utilities.g_val_num.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.KEYVAL_NUM;
END;
```

See Also:

"SAVEKEY_NUM Function (page 28-89)"

28.88 KEYVAL_VC2 Function

This function gets the value of the package variable (`apex_utilities.g_val_vc2`) set by `APEX_UTIL.SAVEKEY_VC2`.

Syntax

```
APEX_UTIL.KEYVAL_VC2;
```

Parameters

None.

Example

The following example shows how to use the `KEYVAL_VC2` function to return the current value of the package variable `apex_utilities.g_val_vc2`.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.KEYVAL_VC2;
END;
```

See Also:

"SAVEKEY_VC2 Function (page 28-90)"

28.89 LOCK_ACCOUNT Procedure

Sets a user account status to locked. Must be run by an authenticated workspace administrator in the context of a page request.

Syntax

```
APEX_UTIL.LOCK_ACCOUNT (
     p_user_name IN VARCHAR2);
```

Parameters

Table 28-73 LOCK_ACCOUNT Parameters

Parameter	Description
`p_user_name`	The user name of the user account.

Example

The following example shows how to use the `LOCK_ACCOUNT` procedure. Use this procedure to lock an Application Express account (workspace administrator,

developer, or end user) in the current workspace. This action locks the account for use by administrators, developers, and end users.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.LOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now locked.');
    END LOOP;
END;
```

See Also:

- "UNLOCK_ACCOUNT Procedure (page 28-123)"
- "GET_ACCOUNT_LOCKED_STATUS Function (page 28-41)"

28.90 PASSWORD_FIRST_USE_OCCURRED Function

Returns TRUE if the account's password has changed since the account was created, an Oracle Application Express administrator performs a password reset operation that results in a new password being emailed to the account holder, or a user has initiated password reset operation. This function returns FALSE if the account's password has not been changed since either of the events just described.

This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.PASSWORD_FIRST_USE_OCCURRED (
    p_user_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 28-74 PASSWORD_FIRST_USE_OCCURRED Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the PASSWORD_FIRST_USE_OCCURRED function. Use this function to check if the password for an Application Express user account (workspace administrator, developer, or end user) in the current workspace has been changed by the user the first time the user logged in after the password was initially set during account creation, or was changed by one of the password reset operations described above.This is meaningful only with accounts for which the CHANGE_PASSWORD_ON_FIRST_USE attribute is set to **Yes**.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        IF APEX_UTIL.PASSWORD_FIRST_USE_OCCURRED(p_user_name => c1.user_name) THEN
            htp.p('User:'||c1.user_name||' has logged in and updated the password.');
        END IF;
    END LOOP;
END;
```

See Also:

"CHANGE_PASSWORD_ON_FIRST_USE Function (page 28-9)"

28.91 PREPARE_URL Function

The PREPARE_URL function serves two purposes:

1. To return an f?p URL with the Session State Protection checksum argument (&cs=) if one is required.
2. To return an f?p URL with the session ID component replaced with zero (0) if the zero session ID feature is in use and other criteria are met.

Note:

The `PREPARE_URL` functions returns the f?p URL with `&cs=<large hex value>` appended. If you use this returned value, for example in JavaScript, it may be necessary to escape the ampersand in the URL to conform with syntax rules of the particular context. One place you may encounter this is in SVG chart SQL queries which might include `PREPARE_URL` calls.

Syntax

```
APEX_UTIL.PREPARE_URL (
    p_url           IN VARCHAR2,
    p_url_charset   IN VARCHAR2 default null,
    p_checksum_type IN VARCHAR2 default null,
    p_triggering_element IN VARCHAR2 default 'this')
RETURN VARCHAR2;
```

Parameters

Table 28-75 PREPARE_URL Parameters

Parameter	Description
`p_url`	An f?p relative URL with all substitutions resolved.
`p_url_charset`	The character set name (for example, `UTF-8`) to use when escaping special characters contained within argument values.
`p_checksum_type`	Null or any of the following six values, `SESSION` or `3`, `PRIVATE_BOOKMARK` or `2`, or `PUBLIC_BOOKMARK` or `1`.
`p_triggering_element`	A jQuery selector (for example, `#my_button` , where `my_button` is the static ID for a button element), to identify which element to use to trigger the dialog. This is required for Modal Dialog support.

Example 1

The following example shows how to use the `PREPARE_URL` function to return a URL with a valid 'SESSION' level checksum argument. This URL sets the value of `P1_ITEM` page item to `xyz`.

```
DECLARE
    l_url varchar2(2000);
    l_app number := v('APP_ID');
    l_session number := v('APP_SESSION');
BEGIN
    l_url := APEX_UTIL.PREPARE_URL(
        p_url => 'f?p=' || l_app || ':1:'||l_session||'::NO::P1_ITEM:xyz',
        p_checksum_type => 'SESSION');
END;
```

Example 2

The following example shows how to use the PREPARE_URL function to return a URL with a zero session ID. In a PL/SQL Dynamic Content region that generates `f?p` URLs (anchors), call PREPARE_URL to ensure that the session ID is set to zero when the zero session ID feature is in use, when the user is a public user (not authenticated), and when the target page is a public page in the current application:

```
htp.p(APEX_UTIL.PREPARE_URL(p_url => 'f?p=' || :APP_ID || ':10:'|| :APP_SESSION
||'::NO::P10_ITEM:ABC');
```

When using PREPARE_URL for this purpose, the `p_url_charset` and `p_checksum_type` arguments can be omitted. However, it is permissible to use them when both the Session State Protection and Zero Session ID features are applicable.

See Also:

"Facilitating Bookmarks by Using Zero as the Session ID"

28.92 PUBLIC_CHECK_AUTHORIZATION Function [DEPRECATED]

Note:

Use the "IS_AUTHORIZED Function (page 4-2)" instead of this deprecated function.

Given the name of a authorization scheme, this function determines if the current user passes the security check.

Syntax

```
APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION (
    p_security_scheme    IN    VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 28-76 PUBLIC_CHECK_AUTHORIZATION Parameters

Parameter	Description
p_security_name	The name of the authorization scheme that determines if the user passes the security check.

Example

The following example shows how to use the PUBLIC_CHECK_AUTHORIZATION function to check if the current user passes the check defined in the my_auth_scheme authorization scheme.

```
DECLARE
    l_check_security  BOOLEAN;
BEGIN
    l_check_security := APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION('my_auth_scheme');
END;
```

28.93 PURGE_REGIONS_BY_APP Procedure

Deletes all cached regions for an application.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_APP (
    p_application IN NUMBER);
```

Parameters

Table 28-77 PURGE_REGIONS_BY_APP Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

The following example show how to use APEX_UTIL.PURGE_REGIONS_BY_APP to delete all cached regions for application #123.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_APP(p_application=>123);
END;
```

28.94 PURGE_REGIONS_BY_NAME Procedure

Deletes all cached values for a region identified by the application ID, page number and region name.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_NAME (
     p_application IN NUMBER,
     p_page        IN NUMBER,
     p_region_name IN VARCHAR2);
```

Parameters

Table 28-78 PURGE_REGIONS_BY_NAME Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Table 28-78 (Cont.) PURGE_REGIONS_BY_NAME Parameters

Parameter	Description
p_page	The number of the page containing the region to be deleted.
p_region_name	The region name to be deleted.

Example

The following example shows how to use the PURGE_REGIONS_BY_NAME procedure to delete all the cached values for the region 'my_cached_region' on page 1 of the current application.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_NAME(
        p_application => :APP_ID,
        p_page => 1,
        p_region_name => 'my_cached_region');
END;
```

28.95 PURGE_REGIONS_BY_PAGE Procedure

Deletes all cached regions by application and page.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_PAGE (
     p_application IN NUMBER,
     p_page        IN NUMBER);
```

Parameters

Table 28-79 PURGE_REGIONS_BY_PAGE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The identification number of page containing the region.

Example

The following example shows how to use the PURGE_REGIONS_BY_PAGE procedure to delete all the cached values for regions on page 1 of the current application.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_PAGE(
        p_application => :APP_ID,
        p_page => 1);
END;
```

28.96 REDIRECT_URL Procedure

This procedure calls owa_util.redirect_url to tell the browser to redirect to a new URL. Afterwards, it automatically calls

apex_application.stop_apex_engine to abort further processing of the Application Express application.

Syntax

```
APEX_UTIL.REDIRECT_URL (
    p_url              in varchar2,
    p_reset_htp_buffer in boolean default true );
```

Parameters

Table 28-80 REDIRECT_URL Parameters

Parameter	Description
p_url	The URL the browser requests.
p_reset_htp_buffer	Set to TRUE to reset the HTP buffer to make sure the browser understands the redirect to the new URL and is not confused by data that is already written to the HTP buffer. Set to FALSE if the application has it's own cookie to use in the response.

Example

The following example tells the browser to redirect to http://www.oracle.com and immediately stops further processing.

```
apex_util.redirect_url (
    p_url => 'http://www.oracle.com/' );
```

28.97 REMOVE_PREFERENCE Procedure

This procedure removes the preference for the supplied user.

Syntax

```
APEX_UTIL.REMOVE_PREFERENCE(
    p_preference    IN    VARCHAR2 DEFAULT NULL,
    p_user          IN    VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 28-81 REMOVE_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference to remove.
p_user	User for whom the preference is defined.

Example

The following example shows how to use the REMOVE_PREFERENCE procedure to remove the preference default_view for the currently authenticated user.

```
BEGIN
    APEX_UTIL.REMOVE_PREFERENCE(
        p_preference => 'default_view',
```

```
        p_user       => :APP_USER);
END;
```

See Also:

- "GET_PREFERENCE Function (page 28-57)"
- "SET_PREFERENCE Procedure (page 28-102)"
- "Managing User Preferences" in *Oracle Application Express Administration Guide*

28.98 REMOVE_SORT_PREFERENCES Procedure

This procedure removes the user's column heading sorting preference value.

Syntax

```
APEX_UTIL.REMOVE_SORT_PREFERENCES (
    p_user  IN   VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 28-82 REMOVE_SORT_PREFERENCES Parameters

Parameter	Description
p_user	Identifies the user for whom sorting preferences are removed.

Example

The following example shows how to use the REMOVE_SORT_PREFERENCES procedure to remove the currently authenticated user's column heading sorting preferences.

```
BEGIN
    APEX_UTIL.REMOVE_SORT_PREFERENCES(:APP_USER);
END;
```

28.99 REMOVE_USER Procedure

This procedure removes the user account identified by the primary key or a user name. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.REMOVE_USER(
    p_user_id   IN NUMBER,
    p_user_name IN VARCHAR2);
```

Parameters

Table 28-83 REMOVE_USER Parameters

Parameter	Description
p_user_id	The numeric primary key of the user account record.
p_user_name	The user name of the user account.

Example

The following examples show how to use the REMOVE_USER procedure to remove a user account. Firstly, by the primary key (using the p_user_id parameter) and secondly by user name (using the p_user_name parameter).

```
BEGIN
    APEX_UTIL.REMOVE_USER(p_user_id=> 99997);
END;

BEGIN
    APEX_UTIL.REMOVE_USER(p_user_name => 'FRANK');
END;
```

28.100 RESET_AUTHORIZATIONS Procedure [DEPRECATED]

Note:

Use the "RESET_CACHE Procedure (page 4-3)" instead of this deprecated procedure.

To increase performance, Oracle Application Express caches the results of authorization schemes after they have been evaluated. You can use this procedure to undo caching, requiring each authorization scheme be revalidated when it is next encountered during page show or accept processing. You can use this procedure if you want users to have the ability to change their responsibilities (their authorization profile) within your application.

Syntax

```
APEX_UTIL.RESET_AUTHORIZATIONS;
```

Parameters

None.

Example

The following example shows how to use the RESET_AUTHORIZATIONS procedure to clear the authorization scheme cache.

```
BEGIN
    APEX_UTIL.RESET_AUTHORIZATIONS;
END;
```

28.101 RESET_PASSWORD Procedure

This procedure is used to change the password of a given user name for the current workspace. This procedure changes the password of `p_user_name` in the current workspace to `p_new_password`. If `p_change_password_on_first_use` is TRUE, then the user has to change the password on the next login.

Syntax

```
APEX_UTIL.RESET_PASSWORD (
    p_user_name      IN VARCHAR2 DEFAULT WWW_FLOW_SECURITY.G_USER,
    p_old_password   IN VARCHAR2 DEFAULT NULL,
    p_new_password   IN VARCHAR2,
    p_change_password_on_first_use IN BOOLEAN DEFAULT TRUE );
```

Parameters

Table 28-84 RESET_PASSWORD Parameters

Parameter	Description
`p_user_name`	The user whose password should be changed. The default is the currently logged in Application Express user name.
`p_old_password`	The current password of the user. The call succeeds if the given value matches the current password or it is null and the owner of the calling PL/SQL code has APEX_ADMINISTRATOR_ROLE. If the value is not the user's password, an error occurs.
`p_new_password`	The new password.
`p_change_password_on_first_use`	If `TRUE` (default), the user must change the password on the next login.

Error Returns

Table 28-85 RESET_PASSWORD Parameters

Error	Description
INVALID_CREDENTIALS	Occurs if `p_user_name` does not match `p_old_password`,
`APEX.AUTHENTICATION.LOGIN_THROTTLE.COUNTER`	Indicates authentication prevented by login throttle.
`internal error`	Occurs if `p_old_password` is `NULL` and caller does not have `APEX_ADMINISTRATOR_ROLE`.
`internal error`	Indicates caller is not a valid workspace schema.

Example

This example demonstrates changing the password of the currently logged in user to a new password.

```
apex_util.reset_password (
    p_old_password => :P111_OLD_PASSWORD,
    p_new_password => :P111_NEW_PASSWORD );
```

28.102 RESET_PW Procedure

This procedure resets the password for a named user and emails it in a message to the email address located for the named account in the current workspace. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.RESET_PW(
    p_user IN VARCHAR2,
    p_msg  IN VARCHAR2);
```

Parameters

Table 28-86 RESET_PW Parameters

Parameter	Description
p_user	The user name of the user account.
p_msg	Message text to be mailed to a user.

Example

The following example shows how to use the RESET_PW procedure to reset the password for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.RESET_PW(
        p_user => 'FRANK',
        p_msg => 'Contact help desk at 555-1212 with questions');
END;
```

See Also:

"CHANGE_CURRENT_USER_PW Procedure (page 28-9)"

28.103 SAVEKEY_NUM Function

This function sets a package variable (apex_utilities.g_val_num) so that it can be retrieved using the function KEYVAL_NUM.

Syntax

```
APEX_UTIL.SAVEKEY_NUM(
    p_val IN NUMBER)
RETURN NUMBER;
```

Parameters

Table 28-87 SAVEKEY_NUM Parameters

Parameter	Description
p_val	The numeric value to be saved.

Example

The following example shows how to use the SAVEKEY_NUM function to set the apex_utilities.g_val_num package variable to the value of 10.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.SAVEKEY_NUM(p_val => 10);
END;
```

See Also:

"KEYVAL_NUM Function (page 28-78)"

28.104 SAVEKEY_VC2 Function

This function sets a package variable (apex_utilities.g_val_vc2) so that it can be retrieved using the function KEYVAL_VC2.

Syntax

```
APEX_UTIL.SAVEKEY_VC2(
    p_val IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-88 SAVEKEY_VC2 Parameters

Parameter	Description
p_val	The is the VARCHAR2 value to be saved.

Example

The following example shows how to use the SAVEKEY_VC2 function to set the apex_utilities.g_val_vc2 package variable to the value of 'XXX'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.SAVEKEY_VC2(p_val => 'XXX');
END;
```

See Also:

"KEYVAL_VC2 Function (page 28-79)"

28.105 SET_APP_BUILD_STATUS Procedure

This procedure sets the build status of the specified application.

Syntax

```
apex_util.set_app_build_stauts( p_application_id IN NUMBER,
                                p_build_status in VARCHAR2 );
```

Parameters

Table 28-89 SET_APP_BUILD_STATUS Parameters

Parameter	Description
p_application_id	The ID of the application.
p_build_status	The new build status of the application. Values include: • RUN_ONLY - The application can be run but cannot be edited by developers. • RUN_AND_BUILD - The application can be run and can also be edited by developers.

Example

```
begin
    apex_util.set_app_build_status(
        p_application_id => 170,
        p_build_status   => 'RUN_ONLY' );
    commit;
end;
```

28.106 SET_APPLICATION_STATUS Procedure

This procedure changes the status of the application.

Syntax

```
APEX_UTIL.SET_APPLICATION_STATUS(
   p_application_id IN NUMBER,
   p_application_status IN VARCHAR2,
   p_unavailable_value IN VARCHAR2,
   p_restricted_user_list IN VARCHAR2);
```

Parameters

Table 28-90 SET_APPLICATION_STATUS Parameters

Parameter	Description
p_application_id	The Application ID.
p_application_status	New application status . Values include: • AVAILABLE - Application is available with no restrictions. • AVAILABLE_W_EDIT_LINK - Application is available with no restrictions. Developer Toolbar shown to developers . • DEVELOPERS_ONLY - Application only available to developers. • RESTRICTED_ACCESS - Application only available to users in p_restricted_user_list. • UNAVAILABLE - Application unavailable. Message shown in p_unavailable_value. • UNAVAILABLE_PLSQL - Application unavailable. Message shown from PL/SQL block in p_unavailable_value. • UNAVAILABLE_URL - Application unavailable. Redirected to URL provided in p_unavailable_value.
p_unavailable_value	Value used when application is unavailable. This value has different semantics dependent upon value for p_application_status.
p_restricted_user_list	Comma separated list of users permitted to access application, when p_application_status = RESTRICTED_ACCESS.

Examples

```
begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'AVAILABLE' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'AVAILABLE_W_EDIT_LINK' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'DEVELOPERS_ONLY' );
end;
```

```
begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'RESTRICTED_ACCESS',
    p_restricted_user_list => 'xxx.xxx@abc.com' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE',
    p_unavailable_value => 'Application not available, sorry' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE_PLSQL',
    p_unavailable_value => 'sys.htp.p(''Application unavailable, sorry'');' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE_URL',
    p_unavailable_value => 'http://www.xyz.com' );
end;
```

See Also:

"Availability" in *Oracle Application Express App Builder User's Guide*

28.107 SET_ATTRIBUTE Procedure

This procedure sets the value of one of the attribute values (1 through 10) of a user in the Application Express accounts table.

Syntax

```
APEX_UTIL.SET_ATTRIBUTE(
    p_userid           IN NUMBER,
    p_attribute_number IN NUMBER,
    p_attribute_value  IN VARCHAR2);
```

Parameters

Table 28-91 SET_ATTRIBUTE Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_attribute_number	Attribute number in the user record (1 through 10).
p_attribute_value	Value of the attribute located by p_attribute_number to be set in the user record.

Example

The following example shows how to use the SET_ATTRIBUTE procedure to set the number 1 attribute for user 'FRANK' with the value 'foo'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    APEX_UTIL.SET_ATTRIBUTE (
        p_userid => apex_util.get_user_id(p_username => 'FRANK'),
        p_attribute_number => 1,
        p_attribute_value => 'foo');
END;
```

See Also:

"GET_ATTRIBUTE Function (page 28-42)"

28.108 SET_AUTHENTICATION_RESULT Procedure

This procedure can be called from an application's custom authentication function (that is, credentials verification function). The status passed to this procedure is logged in the Login Access Log.

Syntax

```
APEX_UTIL.SET_AUTHENTICATION_RESULT(
    p_code IN NUMBER);
```

Parameters

Table 28-92 SET_AUTHENTICATION_RESULT Parameters

Parameter	Description
p_code	Any numeric value the developer chooses. After this value is set in the session using this procedure, it can be retrieved using the APEX_UTIL.GET_AUTHENTICATION_RESULT function.

Example

One way to use this procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. In this example, no credentials verification is performed, it just demonstrates how text and numeric status values can be registered for logging.Note that the status set using this procedure is visible in the apex_user_access_log view and in the reports on this view available to workspace and site administrators.

```
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is back.');
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
```

```
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

See Also:

- "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*
- "GET_AUTHENTICATION_RESULT Function (page 28-43)"
- "SET_CUSTOM_AUTH_STATUS Procedure (page 28-97)"

28.109 SET_BUILD_OPTION_STATUS Procedure

Use this procedure to change the build option status of a specified application.

Note:

The build option status will be overwritten when the application is upgraded to a new version. To keep the status set via the API, it is necessary to set the build option attribute **On Upgrade Keep Status** to **Yes**.

Syntax

```
apex_util.set_build_option_status(p_application_id IN NUMBER,
                                  p_id IN NUMBER,
                                  p_build_status IN VARCHAR2);
```

Parameters

Table 28-93 SET_BUILD_OPTION_STATUS Parameters

Parameter	Description
p_application_id	The ID of the application that owns the build option under shared components.
p_id	The ID of the build option in the application.
p_build_status	The new status of the build option. Possible values are INCLUDE, EXCLUDE both upper case.

Example

The following example demonstrates how to use the SET_BUILD_OPTION_STATUS procedure to change the current status of build option.

```
BEGIN
APEX_UTIL.SET_BUILD_OPTION_STATUS(
    P_APPLICATION_ID => 101,
    P_ID => 245935500311121039, P_BUILD_STATUS=>'INCLUDE');
```

```
END;
```

28.110 SET_CURRENT_THEME_STYLE Procedure [DEPRECATED]

This procedure sets the user interface theme style for an application. For example, if there are more than one theme styles available for the current theme, you can use this procedure to change the application theme style.

Syntax

```
APEX_UTIL.SET_CURRENT_THEME_STYLE(
    p_theme_number IN NUMBER,
    p_theme_style_id IN NUMBER
    );
```

Parameters

Table 28-94 SET_CURRENT_THEME_STYLE Parameters

Parameter	Description
p_theme_number	The current theme number of the application. This can be retrieved from APEX_APPLICATION_THEMES view.
p_theme_style_id	The numeric ID of theme style. You can get available theme styles for an application from APEX_APPLICATION_THEME_STYLES view.

Example

The following example shows how to use the SET_CURRENT_THEME_STYLE procedure to set the current application desktop theme style to Blue.

```
DECLARE
    l_current_theme_number number;
    l_theme_style_id       number;

BEGIN
    select theme_number
    into l_current_theme_number
    from apex_application_themes
    where application_id = :app_id
    and ui_type_name   = 'DESKTOP'
    and is_current = 'Yes';

    select s.theme_style_id
    into l_new_theme_style_id
    from apex_application_theme_styles s, apex_application_themes t
    where s.application_id = t.application_id
    and s.theme_number = t.theme_number
    and s.application_id = :app_id
    and t.ui_type_name   = 'DESKTOP'
    and t.is_current = 'Yes'
    and s.name = 'Blue';

    if l_current_theme_number is not null and
    l_new_theme_style_id is not null then
        APEX_UTIL.SET_CURRENT_THEME_STYLE(
            p_theme_number   => l_current_theme_number,
```

```
            p_theme_style_id => l_new_theme_style_id
            );

    end if;

END;
```

See Also:

"SET_CURRENT_STYLE Procedure (page 26-4)"

28.111 SET_CUSTOM_AUTH_STATUS Procedure

This procedure can be called from an application's custom authentication function (that is, credentials verification function). The status passed to this procedure is logged in the Login Access Log.

Syntax

```
APEX_UTIL.SET_CUSTOM_AUTH_STATUS(
    p_status  IN VARCHAR2);
```

Parameters

Table 28-95 SET_CUSTOM_AUTH_STATUS Parameters

Parameter	Description
p_status	Any text the developer chooses to denote the result of the authentication attempt (up to 4000 characters).

Example

One way to use the SET_CUSTOM_AUTH_STATUS procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. Note that no credentials verification is performed. The status set using this procedure is visible in the apex_user_access_log view and in the reports on this view available to workspace and site administrators.

```
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is back.');
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

See Also:

- "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*
- "SET_AUTHENTICATION_RESULT Procedure (page 28-94)"
- "GET_AUTHENTICATION_RESULT Function (page 28-43)"

28.112 SET_EDITION Procedure

This procedure sets the name of the edition to be used in all application SQL parsed in the current page view or page submission.

Syntax

```
APEX_UTIL.SET_EDITION(
    p_edition IN VARCHAR2);
```

Parameters

Table 28-96 SET_EDITION Parameters

Parameter	Description
p_edition	Edition name.

Example

The following example shows how to use the SET_EDITION procedure. It sets the edition name for the database session of the current page view.

```
BEGIN
    APEX_UTIL.SET_EDITION( P_EDITION => 'Edition1' );
END;
```

Note:

Support for Edition-Based Redefinition is only available in database version 11.2.0.1 or higher.

28.113 SET_EMAIL Procedure

This procedure updates a user account with a new email address. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_EMAIL(
    p_userid IN NUMBER,
    p_email  IN VARCHAR2);
```

Parameters

Table 28-97 SET_EMAIL Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_email	The email address to be saved in user account.

Example

The following example shows how to use the SET_EMAIL procedure to set the value of EMAIL to 'frank.scott@somewhere.com' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_EMAIL(
        p_userid  => APEX_UTIL.GET_USER_ID('FRANK'),
        p_email   => 'frank.scott@somewhere.com');
END;
```

See Also:

- "GET_EMAIL Function (page 28-47)"
- "GET_USER_ID Function (page 28-67)"

28.114 SET_FIRST_NAME Procedure

This procedure updates a user account with a new FIRST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_FIRST_NAME(
    p_userid      IN NUMBER,
    p_first_name  IN VARCHAR2);
```

Parameters

Table 28-98 SET_FIRST_NAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_first_name	FIRST_NAME value to be saved in user account.

Example

The following example shows how to use the SET_FIRST_NAME procedure to set the value of FIRST_NAME to 'FRANK' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_FIRST_NAME(
        p_userid       => APEX_UTIL.GET_USER_ID('FRANK'),
```

```
        p_first_name   => 'FRANK');
END;
```

See Also:

- "GET_FIRST_NAME Function (page 28-50)"
- "GET_USER_ID Function (page 28-67)"

28.115 SET_GLOBAL_NOTIFICATION Procedure

This procedure is used to set the global notification message which is the message displayed in page #GLOBAL_NOTIFICATION# substitution string.

Syntax

```
APEX_UTIL.SET_GLOBAL_NOTIFICATION(
   p_application_id IN NUMBER,
   p_global_notification_message IN VARCHAR2);
```

Parameters

Table 28-99 SET_GLOBAL_NOTIFICATION Parameters

Parameter	Description
p_application_id	The Application ID.
p_global_notification_message	Text string to be used for the global notification message.

Example

```
begin
    apex_util.set_global_notification(
        p_application_id              => 117,
        p_global_notification_message => 'This application will be upgraded this
weekend at 2100 UTC' );
end;
```

See Also:

"Availability" in *Oracle Application Express App Builder User's Guide*

28.116 SET_GROUP_GROUP_GRANTS Procedure

This procedure modifies the group grants for a given group.

Syntax

```
APEX_UTIL.SET_GROUP_GROUP_GRANTS (
    p_group_name IN VARCHAR2,
    p_granted_group_names IN apex_t_varchar2 );
```

Parameters

Table 28-100 SET_GROUP_GROUP_GRANTS Procedure Parameters

Parameter	Description
p_group_name	The target group name.
p_granted_group_names	The names of groups to grant to p_group_name.

Example

This example creates three groups (ACCTS_PAY, ACCTS_REC, MANAGER) and then grants ACCTS_PAY and ACCTS_REC to MANAGER.

```
apex_util.create_user_group (
    p_group_name => 'ACCTS_PAY' );
apex_util.create_user_group (
    p_group_name => 'ACCTS_REC' );
apex_util.create_user_group (
    p_group_name => 'MANAGER' );
apex_util.set_group_group_grants (
    p_group_name => 'MANAGER',
    p_granted_group_names => apex_t_varchar2('ACCTS_PAY', 'ACCTS_REC') );
```

28.117 SET_GROUP_USER_GRANTS Procedure

This procedure modifies the group grants for a given user.

Syntax

```
APEX_UTIL.SET_GROUP_USER_GRANTS (
    p_user_name IN VARCHAR2,
    p_granted_group_names IN apex_t_varchar2 );
```

Parameters

Table 28-101 SET_GROUP_USER_GRANTS Procedure Parameters

Parameter	Description
p_user_name	The target user name.
p_granted_group_names	The names of groups to grant to p_user_name.

Example

This example creates a user group (MANAGER) and a user (Example User) and then grants MANAGER to Example User.

```
apex_util.create_user_group (
    p_group_name => 'MANAGER' );
apex_util.create_user (
    p_user_name => 'Example User',
    p_web_password => 1_random_password );
-- grant MANAGER to Example User
apex_util.set_group_user_grants (
```

```
    p_user_name => 'Example User',
    p_granted_group_names => apex_t_varchar2('MANAGER') );
```

28.118 SET_LAST_NAME Procedure

This procedure updates a user account with a new LAST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_LAST_NAME(
    p_userid      IN NUMBER,
    p_last_name   IN VARCHAR2);
```

Parameters

Table 28-102 SET_LAST_NAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_last_name	LAST_NAME value to be saved in the user account.

Example

The following example shows how to use the SET_LAST_NAME procedure to set the value of LAST_NAME to 'SMITH' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_LAST_NAME(
        p_userid       => APEX_UTIL.GET_USER_ID('FRANK'),
        p_last_name    => 'SMITH');
END;
```

See Also:

- "GET_LAST_NAME Function (page 28-55)"
- "GET_USER_ID Function (page 28-67)"

28.119 SET_PREFERENCE Procedure

This procedure sets a preference that persists beyond the user's current session.

Syntax

```
APEX_UTIL.SET_PREFERENCE (
    p_preference   IN    VARCHAR2 DEFAULT NULL,
    p_value        IN    VARCHAR2 DEFAULT NULL,
    p_user         IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-103 SET_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference (case-sensitive).
p_value	Value of the preference.
p_user	User for whom the preference is being set.

Example

The following example shows how to use the SET_PREFERENCE procedure to set a preference called 'default_view' to the value 'WEEKLY' that persists beyond session for the currently authenticated user.

```
BEGIN
    APEX_UTIL.SET_PREFERENCE(
        p_preference => 'default_view',
        p_value      => 'WEEKLY',
        p_user       => :APP_USER);
END;
```

See Also:

- "GET_PREFERENCE Function (page 28-57)"
- "REMOVE_PREFERENCE Procedure (page 28-85)"

28.120 SET_SECURITY_GROUP_ID Procedure

Use this procedure with apex_util.find_security_group_id to ease the use of the mail package in batch mode. This procedure is especially useful when a schema is associated with more than one workspace. For example, you might want to create a procedure that is run by a nightly job to email all outstanding tasks.

Syntax

```
APEX_UTIL.SET_SECURITY_GROUP_ID (
    p_security_group_id  IN NUMBER);
```

Parameters

Table 28-104 SET_SECURITY_GROUP_ID Parameters

Parameter	Description
p_security_group_id	This is the security group id of the workspace you are working in.

Example

The following example sends an alert to each user that has had a task assigned within the last day.

```
create or replace procedure new_tasks
is
    l_workspace_id      number;
    l_subject           varchar2(2000);
    l_body              clob;
    l_body_html         clob;
begin
    l_workspace_id := apex_util.find_security_group_id (p_workspace => 'PROJECTS');
    apex_util.set_security_group_id (p_security_group_id => l_workspace_id);

    l_body := ' ';
    l_subject := 'You have new tasks';
    for c1 in (select distinct(p.email_address) email_address, p.user_id
                 from teamsp_user_profile p, teamsp_tasks t
                where p.user_id = t.assigned_to_user_id
                  and t.created_on > sysdate - 1
                  and p.email_address is not null ) loop
        l_body_html := '<p />The following tasks have been added.';
        for c2 in (select task_name, due_date
                     from teamsp_tasks
                    where assigned_to_user_id = c1.user_id
                      and created_on > sysdate - 1 ) loop
            l_body_html := l_body_html || '<p />Task: '||c2.task_name||', due '||
c2.due_date;
        end loop;
apex_mail.send (
            p_to        => c1.email_address,
            p_from      => c1.email_address,
            p_body      => l_body,
            p_body_html => l_body_html,
            p_subj      => l_subject );
    end loop;
end;
```

28.121 SET_SESSION_HIGH_CONTRAST_OFF Procedure

This procedure switches off high contrast mode for the current session.

Syntax

```
APEX_UTIL.SET_SESSION_HIGH_CONTRAST_OFF;
```

Parameters

None.

Example

In this example, high contrast mode is switched off for the current session.

```
BEGIN
    apex_util.set_session_high_contrast_off;
END;
```

28.122 SET_SESSION_HIGH_CONTRAST_ON Procedure

This procedure switches on high contrast mode for the current session.

Syntax

```
APEX_UTIL.SET_SESSION_HIGH_CONTRAST_ON;
```

Parameters

None.

Example

In this example, the current session is put into high contrast mode.

```
BEGIN
    apex_util.set_session_high_contrast_on;
END;
```

28.123 SET_SESSION_LANG Procedure

This procedure sets the language to be used for the current user in the current Application Express session. The language must be a valid IANA language name.

Syntax

```
APEX_UTIL.SET_SESSION_LANG(
    p_lang IN VARCHAR2);
```

Parameters

Table 28-105 SET_SESSION_LANG Parameters

Parameter	Description
p_lang	This is an IANA language code. Some examples include: en, de, de-at, zh-cn, and pt-br.

Example

The following example shows how to use the SET_SESSION_LANG procedure. It sets the language for the current user for the duration of the Application Express session.

```
BEGIN
    APEX_UTIL.SET_SESSION_LANG( P_LANG => 'en');
END;
```

28.124 SET_SESSION_LIFETIME_SECONDS Procedure

This procedure sets the current session's Maximum Session Length in Seconds value. overriding the corresponding application attribute. This allows developers to dynamically shorten or lengthen the session life based on criteria determined after the user authenticates.

Syntax

```
APEX_UTIL.SET_SESSION_LIFETIME_SECONDS (
    p_seconds  IN    NUMEBER,
    p_scope    IN    VARCHAR2 DEFAULT 'SESSION');
```

Parameters

Table 28-106 SET_SESSION_LIFETIME_SECONDS Parameters

Parameter	Description
p_seconds	A positive integer indicating the number of seconds the session used by this application is allowed to exist.
p_scope	This parameter is obsolete. The procedure always sets the lifetime for the whole session.

Example 1

The following example shows how to use the SET_SESSION_LIFETIME_SECONDS procedure to set the current application's Maximum Session Length in Seconds attribute to 7200 seconds (two hours).

By allowing the p_scope input parameter to use the default value of 'SESSION', the following example would actually apply to all applications using the current session. This would be the most common use case when multiple Application Express applications use a common authentication scheme and are designed to operate as a suite in a common session.

```
BEGIN
   APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(p_seconds => 7200);
END;
```

Example 2

The following example shows how to use the SET_SESSION_LIFETIME_SECONDS procedure to set the current application's Maximum Session Length in Seconds attribute to 3600 seconds (one hour).

```
BEGIN
    APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(p_seconds => 3600);
END;
```

28.125 SET_SESSION_MAX_IDLE_SECONDS Procedure

Sets the current application's Maximum Session Idle Time in Seconds value for the current session, overriding the corresponding application attribute. This allows developers to dynamically shorten or lengthen the maximum idle time allowed between page requests based on criteria determined after the user authenticates.

Syntax

```
APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS (
    p_seconds  IN    NUMEBER,
    p_scope    IN    VARCHAR2 DEFAULT 'SESSION');
```

Parameters

Table 28-107 SET_SESSION_MAX_IDLE_SECONDS Parameters

Parameter	Description
p_seconds	A positive integer indicating the number of seconds allowed between page requests.
p_scope	This parameter is obsolete. The procedure always sets the lifetime for the whole session

Example 1

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 1200 seconds (twenty minutes). The following example applies to all applications using the current session.

```
BEGIN
   APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 1200);
END;
```

Example 2

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 600 seconds (ten minutes). This example applies to all applications using the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 600);
END;
```

28.126 SET_SESSION_SCREEN_READER_OFF Procedure

This procedure switches off screen reader mode for the current session.

Syntax

```
APEX_UTIL.SET_SESSION_SCREEN_READER_OFF;
```

Parameters

None

Example

In this example, the current session is put into standard mode.

```
BEGIN
    apex_util.set_session_screen_reader_off;
END;
```

28.127 SET_SESSION_SCREEN_READER_ON Procedure

This procedure puts the current session into screen reader mode.

Syntax

```
APEX_UTIL.SET_SESSION_SCREEN_READER_ON;
```

Parameters

None

Example

In this example, the current session is put into screen reader mode.

```
BEGIN
    apex_util.set_session_screen_reader_on;
END;
```

28.128 SET_SESSION_STATE Procedure

This procedure sets session state for a current Oracle Application Express session.

Syntax

```
APEX_UTIL.SET_SESSION_STATE (
    p_name     IN    VARCHAR2 DEFAULT NULL,
    p_value    IN    VARCHAR2 DEFAULT NULL);
    p_commit   IN    BOOLEAN  DEFAULT TRUE);
```

Parameters

Table 28-108 SET_SESSION_STATE Parameters

Parameter	Description
p_name	Name of the application-level or page-level item for which you are setting sessions state.
p_value	Value of session state to set.
p_commit	If true (the default), commit after modifying session state. If false or if the existing value in session state equals p_value, no commit is issued.

Example

The following example shows how to use the SET_SESSION_STATE procedure to set the value of the item 'my_item' to 'myvalue' in the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_STATE('my_item','myvalue');
END;
```

See Also:

- "GET_SESSION_STATE Function (page 28-62)"
- "GET_NUMERIC_SESSION_STATE Function (page 28-56)"
- "Understanding Session State Management" in *Oracle Application Express App Builder User's Guide*

28.129 SET_SESSION_TERRITORY Procedure

This procedure sets the territory to be used for the current user in the current Application Express session. The territory name must be a valid Oracle territory.

Syntax

```
APEX_UTIL.SET_SESSION_TERRITORY(
    p_territory IN VARCHAR2);
```

Parameters

Table 28-109 SET_SESSION_TERRITORY Parameters

Parameter	Description
p_territory	A valid Oracle territory name. Examples include: AMERICA, UNITED KINGDOM, ISRAEL, AUSTRIA, and UNITED ARAB EMIRATES.

Example

The following example shows how to use the SET_SESSION_TERRITORY procedure. It sets the territory for the current user for the duration of the Application Express session.

```
BEGIN
    APEX_UTIL.SET_SESSION_TERRITORY( P_TERRITORY => 'UNITED KINGDOM');
END;
```

28.130 SET_SESSION_TIME_ZONE Procedure

This procedure sets the time zone to be used for the current user in the current Application Express session.

Syntax

```
APEX_UTIL.SET_SESSION_TIME_ZONE(
    p_time_zone IN VARCHAR2);
```

Parameters

Table 28-110 SET_SESSION_TIME_ZONE Parameters

Parameter	Description
p_timezone	A time zone value in the form of hours and minutes. Examples include: +09:00, 04:00, -05:00.

Example

The following example shows how to use the SET_SESSION_TIME_ZONE procedure. It sets the time zone for the current user for the duration of the Application Express session.

```
BEGIN
    APEX_UTIL.SET_SESSION_TIME_ZONE( P_TIME_ZONE => '-05:00');
END;
```

28.131 SET_USERNAME Procedure

This procedure updates a user account with a new USER_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_USERNAME(
    p_userid   IN NUMBER,
    p_username IN VARCHAR2);
```

Parameters

Table 28-111 SET_USERNAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_username	USER_NAME value to be saved in the user account.

Example

The following example shows how to use the SET_USERNAME procedure to set the value of USERNAME to 'USER-XRAY' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_USERNAME(
        p_userid     => APEX_UTIL.GET_USER_ID('FRANK'),
        P_username   => 'USER-XRAY');
END;
```

See Also:

- "GET_USERNAME Function (page 28-68)"
- "GET_USER_ID Function (page 28-67)"

28.132 SET_WORKSPACE_Procedure

This procedure sets the current workspace.

Syntax

```
procedure set_workspace (
    p_workspace in varchar2 );
```

Parameters

Table 28-112 (page 28-111) describes the parameters available in SET_WORKSPACE_Procedure.

Table 28-112 SET_WORKSPACE_Procedure Parameters

Parameters	Description
p_workspace	The workspace's short name.

Example

This example shows how to Set workspace MY_WORKSPACE.

```
apex_util.set_workspace (
    p_workspace => 'MY_WORKSPACE' );
```

28.133 SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure

This procedure displays a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches the high contrast mode on.

Syntax

```
APEX_UTIL.SHOW_HIGH_CONTRAST_MODE_TOGGLE (
    p_on_message  in varchar2 default null,
    p_off_message in varchar2 default null);
```

Parameters

Table 28-113 SHOW_HIGH_CONTRAST_MODE_TOGGLE Parameters

Parameters	Description
p_on_message	Optional text used for the link to switch to high contrast mode, when you are in standard mode. If this parameter is not passed, the default 'Set High Contrast Mode On' text is displayed.
p_off_message	Optional text used for the link to switch to standard mode, when you are in high contrast mode. If this parameter is not passed, the default 'Set High Contrast Mode Off' text is displayed.

Example

When running in standard mode, this procedure displays a link, Set High Contrast Mode On, that when clicked refreshes the current page and switches on high contrast mode. When running in high contrast mode, a link, Set High Contrast Mode Off, is displayed, that refreshes the current page and switches back to standard mode when clicked.

```
BEGIN
    apex_util.show_high_contrast_mode_toggle;
END;
```

Note:

There are also 2 translatable system messages that can be overridden at application level to change the default link text that is returned for this toggle. They include:

- `APEX.SET_HIGH_CONTRAST_MODE_OFF` - Default text = Set High Contrast Mode Off
- `APEX.SET_HIGH_CONTRAST_MODE_ON` - Default text = Set High Contrast Mode On

See Also:

"GET_HIGH_CONTRAST_MODE_TOGGLE Function (page 28-54)"

28.134 SHOW_SCREEN_READER_MODE_TOGGLE Procedure

This procedure displays a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches the screen reader mode on.

Syntax

```
APEX_UTIL.SHOW_SCREEN_READER_MODE_TOGGLE (
    p_on_message  IN VARCHAR2 DEFAULT NULL,
    p_off_message IN VARCHAR2 DEFAULT NULL)
```

Parameters

Table 28-114 SHOW_SCREEN_READER_MODE_TOGGLE Parameters

Parameter	Description
`p_on_message`	Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text is displayed.
`p_off_message`	Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text is displayed.

Example

When running in standard mode, this procedure displays a link 'Set Screen Reader Mode On', that when clicked refreshes the current page and switches on screen reader mode. When running in screen reader mode, a link 'Set Screen Reader Mode Off' is displayed, that when clicked refreshes the current page and switches back to standard mode.

```
BEGIN
    apex_util.show_screen_reader_mode_toggle;
END;
```

28.135 STRING_TO_TABLE Function [DEPRECATED]

Given a string, this function returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2. This array is a VARCHAR2(32767) table.

Syntax

```
APEX_UTIL.STRING_TO_TABLE (
    p_string       IN VARCHAR2,
    p_separator    IN VARCHAR2 DEFAULT ':')
    RETURN APEX_APPLICATION_GLOBAL.VC_ARR2;
```

Parameters

Table 28-115 STRING_TO_TABLE Parameters

Parameter	Description
p_string	String to be converted into a PL/SQL table of type APEX_APPLICATION_GLOBAL.VC_ARR2.
p_separator	String separator. The default is a colon.

Example

The following example shows how to use the STRING_TO_TABLE function. The function is passed the string 'One:Two:Three' in the p_string parameter and it returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2 containing 3 elements, the element at position 1 contains the value 'One', position 2 contains the value 'Two' and position 3 contains the value 'Three'. This is then output using the HTP.P function call.

```
DECLARE
    l_vc_arr2    APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_vc_arr2 := APEX_UTIL.STRING_TO_TABLE('One:Two:Three');
    FOR z IN 1..l_vc_arr2.count LOOP
        htp.p(l_vc_arr2(z));
    END LOOP;
END;
```

See Also:

- "TABLE_TO_STRING Function [DEPRECATED] (page 28-120)"
- SPLIT Function Signature 1 (page 25-12)
- SPLIT Function Signature 2 (page 25-13)
- SPLIT_NUMBERS Function (page 25-14)

28.136 STRONG_PASSWORD_CHECK Procedure

This procedure returns `Boolean OUT` values based on whether a proposed password meets the password strength requirements as defined by the Oracle Application Express site administrator.

Syntax

```
APEX_UTIL.STRONG_PASSWORD_CHECK(
    p_username                    IN  VARCHAR2,
    p_password                    IN  VARCHAR2,
    p_old_password                IN  VARCHAR2,
    p_workspace_name              IN  VARCHAR2,
    p_use_strong_rules            IN  BOOLEAN,
    p_min_length_err              OUT BOOLEAN,
    p_new_differs_by_err          OUT BOOLEAN,
    p_one_alpha_err               OUT BOOLEAN,
    p_one_numeric_err             OUT BOOLEAN,
    p_one_punctuation_err         OUT BOOLEAN,
    p_one_upper_err               OUT BOOLEAN,
    p_one_lower_err               OUT BOOLEAN,
    p_not_like_username_err       OUT BOOLEAN,
    p_not_like_workspace_name_err OUT BOOLEAN,
    p_not_like_words_err          OUT BOOLEAN,
    p_not_reusable_err            OUT BOOLEAN);
```

Parameters

Table 28-116 STRONG_PASSWORD_CHECK Parameters

Parameter	Description
`p_username`	Username that identifies the account in the current workspace.
`p_password`	Password to be checked against password strength rules.
`p_old_password`	Current password for the account. Used only to enforce "new password must differ from old" rule.
`p_workspace_name`	Current workspace name, used only to enforce "password must not contain workspace name" rule.
`p_use_strong_rules`	Pass `FALSE` when calling this API.
`p_min_length_err`	Result returns `TRUE` or `FALSE` depending upon whether the password meets minimum length requirement.

Table 28-116 (Cont.) STRONG_PASSWORD_CHECK Parameters

Parameter	Description
p_new_differs_by_err	Result returns TRUE or FALSE depending upon whether the password meets "new password must differ from old" requirements.
p_one_alpha_err	Result returns TRUE or FALSE depending upon whether the password meets requirement to contain at least one alphabetic character.
p_one_numeric_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one numeric character.
p_one_punctuation_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one punctuation character.
p_one_upper_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one upper-case character.
p_one_lower_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one lower-case character.
p_not_like_username_err	Result returns TRUE or FALSE depending upon whether the password meets requirements that it not contain the username.
p_not_like_workspace_name_err	Result returns TRUE or FALSE whether upon whether the password meets requirements that it not contain the workspace name.
p_not_like_words_err	Result returns TRUE or FALSE whether the password meets requirements that it not contain specified simple words.
p_not_reusable_err	Result returns TRUE or FALSE whether the password can be reused based on password history rules.

Example

The following example shows how to use the STRONG_PASSWORD_CHECK procedure. It checks the new password 'foo' for the user 'SOMEBODY' meets all the password strength requirements defined by the Oracle Application Express site administrator. If any of the checks fail (the associated OUT parameter returns TRUE), then the example outputs a relevant message. For example, if the Oracle Application Express site administrator has defined that passwords must have at least one numeric character and the password 'foo' was checked, then the p_one_numeric_err OUT parameter would return TRUE and the message 'Password must contain at least one numeric character' would be output.

```
DECLARE
    l_username                  varchar2(30);
    l_password                  varchar2(30);
    l_old_password              varchar2(30);
    l_workspace_name            varchar2(30);
    l_min_length_err            boolean;
```

```
    l_new_differs_by_err          boolean;
    l_one_alpha_err               boolean;
    l_one_numeric_err             boolean;
    l_one_punctuation_err         boolean;
    l_one_upper_err               boolean;
    l_one_lower_err               boolean;
    l_not_like_username_err       boolean;
    l_not_like_workspace_name_err boolean;
    l_not_like_words_err          boolean;
    l_not_reusable_err            boolean;
    l_password_history_days       pls_integer;
BEGIN
    l_username := 'SOMEBODY';
    l_password := 'foo';
    l_old_password := 'foo';
    l_workspace_name := 'XYX_WS';
    l_password_history_days :=
        apex_instance_admin.get_parameter ('PASSWORD_HISTORY_DAYS');

    APEX_UTIL.STRONG_PASSWORD_CHECK(
        p_username                    => l_username,
        p_password                    => l_password,
        p_old_password                => l_old_password,
        p_workspace_name              => l_workspace_name,
        p_use_strong_rules            => false,
        p_min_length_err              => l_min_length_err,
        p_new_differs_by_err          => l_new_differs_by_err,
        p_one_alpha_err               => l_one_alpha_err,
        p_one_numeric_err             => l_one_numeric_err,
        p_one_punctuation_err         => l_one_punctuation_err,
        p_one_upper_err               => l_one_upper_err,
        p_one_lower_err               => l_one_lower_err,
        p_not_like_username_err       => l_not_like_username_err,
        p_not_like_workspace_name_err => l_not_like_workspace_name_err,
        p_not_like_words_err          => l_not_like_words_err,
        p_not_reusable_err            => l_not_reusable_err);

    IF l_min_length_err THEN
        htp.p('Password is too short');
    END IF;

    IF l_new_differs_by_err THEN
        htp.p('Password is too similar to the old password');
    END IF;

    IF l_one_alpha_err THEN
        htp.p('Password must contain at least one alphabetic character');
    END IF;

    IF l_one_numeric_err THEN
        htp.p('Password  must contain at least one numeric character');
    END IF;

    IF l_one_punctuation_err THEN
        htp.p('Password  must contain at least one punctuation character');
    END IF;

    IF l_one_upper_err THEN
        htp.p('Password must contain at least one upper-case character');
    END IF;
```

```
    IF l_one_lower_err THEN
        htp.p('Password must contain at least one lower-case character');
    END IF;

    IF l_not_like_username_err THEN
        htp.p('Password may not contain the username');
    END IF;

    IF l_not_like_workspace_name_err THEN
        htp.p('Password may not contain the workspace name');
    END IF;

    IF l_not_like_words_err THEN
        htp.p('Password contains one or more prohibited common words');
    END IF;

    IF l_not_reusable_err THEN
        htp.p('Password cannot be used because it has been used for the account
within the last '||l_password_history_days||' days.');
    END IF;
END;
```

See Also:

"Creating Strong Password Policies" in *Oracle Application Express Administration Guide*.

28.137 STRONG_PASSWORD_VALIDATION Function

This function returns formatted HTML in a VARCHAR2 result based on whether a proposed password meets the password strength requirements as defined by the Oracle Application Express site administrator.

Syntax

```
FUNCTION STRONG_PASSWORD_VALIDATION(
    p_username                     IN  VARCHAR2,
    p_password                     IN  VARCHAR2,
    P_OLD_PASSWORD                 IN  VARCHAR2 DEFAULT NULL,
    P_WORKSPACE_NAME               IN  VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 28-117 STRONG_PASSWORD_VALIDATION Parameters

Parameter	Description
p_username	Username that identifies the account in the current workspace.
p_password	Password to be checked against password strength rules.
p_old_password	Current password for the account. Used only to enforce "new password must differ from old" rule.
p_workspace_name	Current workspace name, used only to enforce "password must not contain workspace name" rule.

Example

The following example shows how to use the STRONG_PASSWORD_VALIDATION procedure. It checks the new password 'foo' for the user 'SOMEBODY' meets all the password strength requirements defined by the Oracle Application Express site administrator. If any of the checks fail, then the example outputs formatted HTML showing details of where the new password fails to meet requirements.

```
DECLARE
      l_username                    varchar2(30);
      l_password                    varchar2(30);
      l_old_password                varchar2(30);
      l_workspace_name              varchar2(30);
BEGIN
    l_username := 'SOMEBODY';
    l_password := 'foo';
    l_old_password := 'foo';
    l_workspace_name := 'XYX_WS';

    HTP.P(APEX_UTIL.STRONG_PASSWORD_VALIDATION(
        p_username                    => l_username,
        p_password                    => l_password,
        p_old_password                => l_old_password,
        p_workspace_name              => l_workspace_name));
END;
```

28.138 SUBMIT_FEEDBACK Procedure

This procedure enables you to write a procedure to submit feedback, rather than using the page that can be generated by create page of type feedback.

Syntax

```
APEX_UTIL.SUBMIT_FEEDBACK (
    p_comment          IN VARCHAR2 DEFAULT NULL,
    p_type             IN NUMBER   DEFAULT '1',
    p_application_id   IN VARCHAR2 DEFAULT NULL,
    p_page_id          IN VARCHAR2 DEFAULT NULL,
    p_email            IN VARCHAR2 DEFAULT NULL,
    p_screen_width     IN VARCHAR2 DEFAULT NULL,
    p_screen_height    IN VARCHAR2 DEFAULT NULL,
    p_attribute_01     IN VARCHAR2 DEFAULT NULL,
    p_attribute_02     IN VARCHAR2 DEFAULT NULL,
    p_attribute_03     IN VARCHAR2 DEFAULT NULL,
    p_attribute_04     IN VARCHAR2 DEFAULT NULL,
    p_attribute_05     IN VARCHAR2 DEFAULT NULL,
    p_attribute_06     IN VARCHAR2 DEFAULT NULL,
    p_attribute_07     IN VARCHAR2 DEFAULT NULL,
    p_attribute_08     IN VARCHAR2 DEFAULT NULL,
    p_label_01         IN VARCHAR2 DEFAULT NULL,
    p_label_02         IN VARCHAR2 DEFAULT NULLl,
    p_label_03         IN VARCHAR2 DEFAULT NULL,
    p_label_04         IN VARCHAR2 DEFAULT NULL,
    p_label_05         IN VARCHAR2 DEFAULT NULL,
    p_label_06         IN VARCHAR2 DEFAULT NULL,
    p_label_07         IN VARCHAR2 DEFAULT NULL,
    p_label_08         IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-118 SUBMIT_FEEDBACK Parameters

Parameter	Description
p_comment	Comment to be submitted.
p_type	Type of feedback (1 is General Comment, 2 is Enhancement Request, 3 is Bug).
p_application_id	ID of application related to the feedback.
p_page_id	ID of page related to the feedback.
p_email	Email of the user providing the feedback.
p_screen_width	Width of screen at time feedback was provided.
p_screen_height	Height of screen at time feedback was provided.
p_attribute_01	Custom attribute for collecting feedback.
p_attribute_02	Custom attribute for collecting feedback.
p_attribute_03	Custom attribute for collecting feedback.
p_attribute_04	Custom attribute for collecting feedback.
p_attribute_05	Custom attribute for collecting feedback.
p_attribute_06	Custom attribute for collecting feedback.
p_attribute_07	Custom attribute for collecting feedback.
p_attribute_08	Custom attribute for collecting feedback.
p_label_01	Label for corresponding custom attribute.
p_label_02	Label for corresponding custom attribute.
p_label_03	Label for corresponding custom attribute.
p_label_04	Label for corresponding custom attribute.
p_label_05	Label for corresponding custom attribute.
p_label_06	Label for corresponding custom attribute.
p_label_07	Label for corresponding custom attribute.
p_label_08	Label for corresponding custom attribute.

Example

The following example submits a bug about page 22 within application 283.

```
begin
    apex_util.submit_feedback (
       p_comment        => 'This page does not render properly for me',
       p_type           => 3,
       p_application_id => 283,
```

```
       p_page_id        => 22,
       p_email          => 'user@xyz.corp',
       p_attribute_01   => 'Charting',
       p_label_01       => 'Component' );
end;
/
```

28.139 SUBMIT_FEEDBACK_FOLLOWUP Procedure

This procedure enables you to submit follow up to a feedback.

Syntax

```
APEX_UTIL.SUBMIT_FEEDBACK_FOLLOWUP (
    p_feedback_id      IN NUMBER,
    p_follow_up        IN VARCHAR2 DEFAULT NULL,
    p_email            IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 28-119 SUBMIT_FEEDBACK_FOLLOWUP Parameters

Parameter	Description
p_feedback_followup	ID of feedback that this is a follow up to.
p_follow_up	Text of follow up.
p_email	Email of user providing the follow up.

Example

The following example submits follow up to a previously filed feedback.

```
begin
   apex_util.submit_feedback_followup (
       p_feedback_id    => 12345,
       p_follow_up      => 'I tried this on another instance and it does not work
there either',
       p_email          => 'user@xyz.corp' );
end;
/
```

28.140 TABLE_TO_STRING Function [DEPRECATED]

Given a a PL/SQL table of type APEX_APPLICATION_GLOBAL.VC_ARR2, this function returns a delimited string separated by the supplied separator, or by the default separator, a colon (:).

Syntax

```
APEX_UTIL.TABLE_TO_STRING (
    p_table       IN     APEX_APPLICATION_GLOBAL.VC_ARR2,
    p_string      IN     VARCHAR2 DEFAULT ':')
RETURN VARCHAR2;
```

Parameters

Table 28-120 TABLE_TO_STRING Parameters

Parameter	Description
p_string	String separator. Default separator is a colon (:).
p_table	PL/SQL table that is to be converted into a delimited string.

Example

The following function returns a comma delimited string of contact names that are associated with the provided `cust_id`.

```
create or replace function get_contacts (
    p_cust_id  in  number )
    return varchar2
is
    l_vc_arr2   apex_application_global.vc_arr2;
    l_contacts  varchar2(32000);
begin

    select contact_name
        bulk collect
        into l_vc_arr2
        from contacts
    where cust_id = p_cust_id
        order by contact_name;

    l_contacts :=  apex_util.table_to_string (
                       p_table => l_vc_arr2,
                       p_string => ', ');

   return l_contacts;

end get_contacts;
```

See Also:

- "STRING_TO_TABLE Function [DEPRECATED] (page 28-113)"
- "JOIN Function Signature 1 (page 25-6)"
- "JOIN Function Signature 2 (page 25-7)"
- "JOIN_CLOB Function (page 25-6)"

28.141 UNEXPIRE_END_USER_ACCOUNT Procedure

Makes expired end users accounts and the associated passwords usable, enabling a end user to log in to developed applications.

Syntax

```
APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT (
    p_user_name IN VARCHAR2);
```

Parameters

Table 28-121 UNEXPIRE_END_USER_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the UNEXPIRE_END_USER_ACCOUNT procedure. Use this procedure to renew (unexpire) an Application Express end user account in the current workspace. This action specifically renews the account for use by end users to authenticate to developed applications and may also renew the account for use by developers or administrators to log in to a workspace.

This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```

See Also:

- "Table 28-23 (page 28-30)"
- "END_USER_ACCOUNT_DAYS_LEFT Function (page 28-29)"

28.142 UNEXPIRE_WORKSPACE_ACCOUNT Procedure

Unexpires developer and workspace administrator accounts and the associated passwords, enabling the developer or administrator to log in to a workspace.

Syntax

```
APEX_UTIL.UNEXPIRE_WORKSPACE_ACCOUNT (
    p_user_name IN VARCHAR2);
```

Parameters

Table 28-122 UNEXPIRE_WORKSPACE_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the UNEXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to renew (unexpire) an Application Express workspace administrator account in the current workspace. This action specifically renews the

account for use by developers or administrators to login to a workspace and may also renew the account for its use by end users to authenticate to developed applications.

This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (select user_name from apex_users) loop
        APEX_UTIL.UNEXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```

See Also:

- "EXPIRE_WORKSPACE_ACCOUNT Procedure (page 28-30)"
- "WORKSPACE_ACCOUNT_DAYS_LEFT Function (page 28-125)"

28.143 UNLOCK_ACCOUNT Procedure

Sets a user account status to unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.UNLOCK_ACCOUNT (
     p_user_name IN VARCHAR2);
```

Parameters

Table 28-123 UNLOCK_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the UNLOCK_ACCOUNT procedure. Use this procedure to unlock an Application Express account in the current workspace. This action unlocks the account for use by administrators, developers, and end users.This procedure must be run by a user who has administration privileges in the current workspace

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.UNLOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now unlocked.');
    END LOOP;
END;
```

See Also:

- "LOCK_ACCOUNT Procedure (page 28-79)"
- "GET_ACCOUNT_LOCKED_STATUS Function (page 28-41)"

28.144 URL_ENCODE Function

The following special characters are encoded as follows:

Special Characters	After Encoding
%	%25
+	%2B
space	+
.	%2E
*	%2A
?	%3F
\	%5C
/	%2F
>	%3E
<	%3C
}	%7B
{	%7D
~	%7E
[	%5B
]	%5D
'	%60
;	%3B
?	%3F
@	%40
&	%26
#	%23
|	%7C
^	%5E
:	%3A
=	%3D
$	%24

Syntax

```
APEX_UTIL.URL_ENCODE (
    p_url   IN    VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

Table 28-124 URL_ENCODE Parameters

Parameter	Description
p_url	The string to be encoded.

Example

The following example shows how to use the URL_ENCODE function.

```
DECLARE
    l_url  VARCHAR2(255);
BEGIN
    l_url := APEX_UTIL.URL_ENCODE('http://www.myurl.com?id=1&cat=foo');
END;
```

In this example, the following URL:

```
http://www.myurl.com?id=1&cat=foo
```

Would be returned as:

```
http%3A%2F%2Fwww%2Emyurl%2Ecom%3Fid%3D1%26cat%3Dfoo
```

28.145 WORKSPACE_ACCOUNT_DAYS_LEFT Function

Returns the number of days remaining before the developer or workspace administrator account password expires. This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT (
    p_user_name IN VARCHAR2)
    RETURN NUMBER;
```

Parameters

Table 28-125 WORKSPACE_ACCOUNT_DAYS_LEFT Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the WORKSPACE_ACCOUNT_DAYS_LEFT function. It can be used in to find the number of days remaining before an Application Express administrator or developer account in the current workspace expires.

```
DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        l_days_left := APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT(p_user_name =>
c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' expires in '||l_days_left||'
days.');
    END LOOP;
END;
```

See Also:

- "EXPIRE_WORKSPACE_ACCOUNT Procedure (page 28-30)"
- "UNEXPIRE_WORKSPACE_ACCOUNT Procedure (page 28-122)"

29

APEX_WEB_SERVICE

The APEX_WEB_SERVICE API enables you to integrate other systems with Application Express by allowing you to interact with Web services anywhere you can use PL/SQL in your application. The API contains procedures and functions to call both SOAP and RESTful style Web services. It contains functions to parse the responses from Web services and to encode/decode into SOAP friendly base64 encoding.This API also contains package globals for managing cookies and HTTP headers when calling Web services whether from the API or by using standard processes of type Web service. Cookies and HTTP headers can be set before invoking a call to a Web service by populating the globals and the cookies and HTTP headers returned from the Web service response can be read from other globals.

29.1 About the APEX_WEB_SERVICE API

Use the APEX_WEB_SERVICE API to invoke a Web service and examine the response anywhere you can use PL/SQL in Application Express.

The following are examples of when you might use the APEX_WEB_SERVICE API:

- When you want to invoke a Web service by using an On Demand Process using Ajax.
- When you want to invoke a Web service as part of an Authentication Scheme.
- When you need to pass a large binary parameter to a Web service that is base64 encoded.
- When you want to invoke a Web service as part of a validation.

29.2 Invoking a SOAP Style Web Service

There is a procedure and a function to invoke a SOAP style Web service. The procedure stores the response in the collection specified by the parameter p_collection_name. The function returns the results as an XMLTYPE. To retrieve a specific value from the response, you use either the PARSE_RESPONSE function if the result is stored in a collection or the PARSE_XML function if the response is returned as an XMLTYPE. To pass a binary parameter to the Web service as base64 encoded character data, use the function BLOB2CLOBBASE64. Conversely, to transform a response that contains a binary parameter that is base64 encoded use the function CLOBBASE642BLOB. The following is an example of using the BLOB2CLOBBASE64 function to encode a parameter, MAKE_REQUEST procedure to call a Web service, and the PARSE_RESPONSE function to extract a specific value from the response.

```
declare
 l_filename varchar2(255);
 l_BLOB BLOB;
 l_CLOB CLOB;
 l_envelope CLOB;
 l_response_msg varchar2(32767);
BEGIN
 IF :P1_FILE IS NOT NULL THEN
    SELECT filename, BLOB_CONTENT
      INTO l_filename, l_BLOB
      FROM APEX_APPLICATION_FILES
      WHERE name = :P1_FILE;

    l_CLOB := apex_web_service.blob2clobbase64(l_BLOB);

   l_envelope := q'!<?xml version='1.0' encoding='UTF-8'?>!';
   l_envelope := l_envelope|| '<soapenv:Envelope xmlns:soapenv="http://
schemas.xmlsoap.org/soap/envelope/" xmlns:chec="http://www.stellent.com/CheckIn/">
  <soapenv:Header/>
  <soapenv:Body>
     <chec:CheckInUniversal>
        <chec:dDocName>'||l_filename||'</chec:dDocName>
        <chec:dDocTitle>'||l_filename||'</chec:dDocTitle>
        <chec:dDocType>Document</chec:dDocType>
        <chec:dDocAuthor>GM</chec:dDocAuthor>
        <chec:dSecurityGroup>Public</chec:dSecurityGroup>
        <chec:dDocAccount></chec:dDocAccount>
```

```
        <chec:CustomDocMetaData>
           <chec:property>
              <chec:name></chec:name>
              <chec:value></chec:value>
           </chec:property>
        </chec:CustomDocMetaData>
        <chec:primaryFile>
           <chec:fileName>'||l_filename'||</chec:fileName>
           <chec:fileContent>'||l_CLOB||'</chec:fileContent>
        </chec:primaryFile>
        <chec:alternateFile>
           <chec:fileName></chec:fileName>
           <chec:fileContent></chec:fileContent>
        </chec:alternateFile>
        <chec:extraProps>
           <chec:property>
              <chec:name></chec:name>
              <chec:value></chec:value>
           </chec:property>
        </chec:extraProps>
     </chec:CheckInUniversal>
  </soapenv:Body>
</soapenv:Envelope>';

apex_web_service.make_request(
   p_url               => 'http://192.0.2.1/idc/idcplg',
   p_action            => 'http://192.0.2.1/CheckIn/',
   p_collection_name   => 'STELLENT_CHECKIN',
   p_envelope          => l_envelope,
   p_username          => 'sysadmin',
   p_password          => 'password' );

 l_response_msg := apex_web_service.parse_response(
  p_collection_name=>'STELLENT_CHECKIN',
p_xpath=>'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/
idc:statusMessage/text()',
  p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');

 :P1_RES_MSG := l_response_msg;

 END IF;
END;
```

29.3 Invoking a RESTful Style Web Service

RESTful style Web services use a simpler architecture than SOAP. Typically the input to a RESTful style Web service is a collection of name/value pairs. The response can be an XML document or simply text such as a comma separated response or JSON. The following is an example of MAKE_REST_REQUEST being used in an application process that is callable by Ajax.

```
declare
  l_clob clob;
  l_buffer         varchar2(32767);
  l_amount         number;
  l_offset         number;
begin

  l_clob := apex_web_service.make_rest_request(
              p_url => 'http://us.music.yahooapis.com/ video/v1/list/published/
popular',
```

```
              p_http_method => 'GET',
              p_parm_name => apex_util.string_to_table('appid:format'),
              p_parm_value => apex_util.string_to_table(apex_application.g_x01||':'||
apex_application.g_x02));

    l_amount := 32000;
    l_offset := 1;
    begin
        loop
            dbms_lob.read( l_clob, l_amount, l_offset, l_buffer );
            htp.p(l_buffer);
            l_offset := l_offset + l_amount;
            l_amount := 32000;
        end loop;
    exception
        when no_data_found then
            null;
    end;

end;
```

29.4 Retrieving Cookies and HTTP Headers

When you invoke a Web service using any of the supported methods in Application Express, the `g_response_cookies` and `g_headers` globals are populated if the Web service response included any cookies or HTTP headers. You can interrogate these globals and store the information in collections.The following are examples of interrogating the `APEX_WEB_SERVICE` globals to store cookie and HTTP header responses in collections.

```
declare
  i number;
  secure varchar2(1);
begin
  apex_collection.create_or_truncate_collection('P31_RESP_COOKIES');
  for i in 1.. apex_web_service.g_response_cookies.count loop
    IF (apex_web_service.g_response_cookies(i).secure) THEN
      secure := 'Y';
    ELSE
      secure := 'N';
    END IF;
    apex_collection.add_member(p_collection_name => 'P31_RESP_COOKIES',
      p_c001 => apex_web_service.g_response_cookies(i).name,
      p_c002 => apex_web_service.g_response_cookies(i).value,
      p_c003 => apex_web_service.g_response_cookies(i).domain,
      p_c004 => apex_web_service.g_response_cookies(i).expire,
      p_c005 => apex_web_service.g_response_cookies(i).path,
      p_c006 => secure,
      p_c007 => apex_web_service.g_response_cookies(i).version );
  end loop;
end;

declare
  i number;
begin
apex_collection.create_or_truncate_collection('P31_RESP_HEADERS');

for i in 1.. apex_web_service.g_headers.count loop
  apex_collection.add_member(p_collection_name => 'P31_RESP_HEADERS',
    p_c001 => apex_web_service.g_headers(i).name,
```

```
    p_c002 => apex_web_service.g_headers(i).value,
    p_c003 => apex_web_service.g_status_code);
end loop;
end;
```

29.5 Setting Cookies and HTTP Headers

You set cookies and HTTP headers that should be sent along with a Web service request by populating the globals `g_request_cookies` and `g_request_headers` before the process that invokes the Web service.The following examples show populating the globals to send cookies and HTTP headers with a request.

```
for c1 in (select seq_id, c001, c002, c003, c004, c005, c006, c007
             from apex_collections
            where collection_name = 'P31_RESP_COOKIES' ) loop
  apex_web_service.g_request_cookies(c1.seq_id).name := c1.c001;
  apex_web_service.g_request_cookies(c1.seq_id).value := c1.c002;
  apex_web_service.g_request_cookies(c1.seq_id).domain := c1.c003;
  apex_web_service.g_request_cookies(c1.seq_id).expire := c1.c004;
  apex_web_service.g_request_cookies(c1.seq_id).path := c1.c005;
  if c1.c006 = 'Y' then
    apex_web_service.g_request_cookies(c1.seq_id).secure := true;
  else
    apex_web_service.g_request_cookies(c1.seq_id).secure := false;
  end if;
  apex_web_service.g_request_cookies(c1.seq_id).version := c1.c007;
end loop;

for c1 in (select seq_id, c001, c002
             from apex_collections
            where collection_name = 'P31_RESP_HEADERS' ) loop
  apex_web_service.g_request_headers(c1.seq_id).name := c1.c001;
  apex_web_service.g_request_headers(c1.seq_id).value := c1.c002;
end loop;
```

29.6 BLOB2CLOBBASE64 Function

Use this function to convert a `BLOB` datatype into a `CLOB` that is `base64` encoded. This is often used when sending a binary as an input to a Web service.

Syntax

```
APEX_WEB_SERVICE.BLOB2CLOBBASE64 (
        p_blob IN BLOB)
RETURN CLOB;
```

Parameters

Table 29-1 BLOB2CLOBBASE64 Parameters

Parameter	Description
`p_blob`	The BLOB to convert into base64 encoded CLOB.

Example

The following example gets a file that was uploaded from the `apex_application_files` view and converts the `BLOB` into a `CLOB` that is `base64` encoded.

```
declare
        l_clob    CLOB;
    l_blob    BLOB;
begin
    SELECT BLOB_CONTENT
      INTO l_BLOB
      FROM APEX_APPLICATION_FILES
      WHERE name = :P1_FILE;

    l_CLOB := apex_web_service.blob2clobbase64(l_BLOB);
end;
```

29.7 CLOBBASE642BLOB Function

Use this function to convert a `CLOB` datatype that is `base64` encoded into a `BLOB`. This is often used when receiving output from a Web service that contains a binary parameter.

Syntax

```
APEX_WEB_SERVICE.CLOBBASE642BLOB (
        p_clob IN CLOB)
RETURN BLOB;
```

Parameters

Table 29-2 CLOBBASE642BLOB Parameters

Parameter	Description
`p_clob`	The `base64` encoded `CLOB` to convert into a `BLOB`.

Example

The following example retrieves a `base64` encoded node from an XML document as a `CLOB` and converts it into a `BLOB`.

```
declare
        l_base64    CLOB;
    l_blob    BLOB;
    l_xml       XMLTYPE;
begin
    l_base64 := apex_web_service.parse_xml_clob(l_xml, ' //runReportReturn/
reportBytes/text()');
        l_blob := apex_web_service.clobbase642blob(l_base64);
end;
```

29.8 MAKE_REQUEST Procedure

Use this procedure to invoke a SOAP style Web service with the supplied SOAP envelope and store the results in a collection.

Syntax

```
APEX_WEB_SERVICE.MAKE_REQUEST (
    p_url               IN VARCHAR2,
    p_action            IN VARCHAR2 default null,
    p_version           IN VARCHAR2 default '1.1',
    p_collection_name   IN VARCHAR2 default null,
    p_envelope          IN CLOB,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_scheme            IN VARCHAR2 DEFAULT 'Basic',
    p_proxy_override    IN VARCHAR2 default null,
    p_transfer_timeout  IN NUMBER   default 180,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null );
```

Parameters

Table 29-3 (page 29-7) describes the parameters available in the MAKE_REQUEST procedure.

Table 29-3 MAKE_REQUEST Procedure Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation to be invoked.
p_version	The SOAP version, 1.1 or 1.2. The default is 1.1.
p_collection_name	The name of the collection to store the response.
p_envelope	The SOAP envelope to post to the service.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme, Basic (default) or AWS or Digest if supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.

Example

The following example uses the make_request procedure to retrieve a list of movies from a SOAP style Web service. The response is stored in an Application Express collection named MOVIE_LISTINGS.

```
declare
     l_envelope CLOB;
BEGIN
    l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <tns:GetTheatersAndMovies>
         <tns:zipCode>43221</tns:zipCode>
         <tns:radius>5</tns:radius>
      </tns:GetTheatersAndMovies>
   </soap:Body>
</soap:Envelope>';

apex_web_service.make_request(
   p_url               => ' http://www.ignyte.com/webservices/
ignyte.whatsshowing.webservice/moviefunctions.asmx',
   p_action            => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
   p_collection_name   => 'MOVIE_LISTINGS',
   p_envelope          => l_envelope
);
END;
```

29.9 MAKE_REQUEST Function

Use this function to invoke a SOAP style Web service with the supplied SOAP envelope returning the results in an XMLTYPE.

Syntax

```
APEX_WEB_SERVICE.MAKE_REQUEST (
    p_url               IN VARCHAR2,
    p_action            IN VARCHAR2 default null,
    p_version           IN VARCHAR2 default '1.1',
    p_envelope          IN CLOB,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_scheme            IN VARCHAR2 default 'Basic',
    p_proxy_override    IN VARCHAR2 default null,
    p_transfer_timeout  IN NUMBER   default 180,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null )
RETURN XMLTYPE;
```

Parameters

Table 29-4 MAKE_REQUEST Function Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation to be invoked.
p_version	The SOAP version, 1.1 or 1.2. The default is 1.1.
p_envelope	The SOAP envelope to post to the service.

Table 29-4 (Cont.) MAKE_REQUEST Function Parameters

Parameter	Description
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CRED if supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.

Example

The following example uses the make_request function to invoke a SOAP style Web service that returns movie listings. The result is stored in an XMLTYPE.

```
declare
    l_envelope CLOB;
    l_xml    XMLTYPE;
BEGIN
        l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
        <tns:GetTheatersAndMovies>
            <tns:zipCode>43221</tns:zipCode>
            <tns:radius>5</tns:radius>
        </tns:GetTheatersAndMovies>
    </soap:Body>
</soap:Envelope>';

l_xml := apex_web_service.make_request(
    p_url => ' http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/
moviefunctions.asmx',
   p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
   p_envelope => l_envelope
);
END
```

29.10 MAKE_REST_REQUEST Function

Use this function to invoke a RESTful style Web service supplying either name value pairs, a character based payload or a binary payload and returning the response in a CLOB.

Syntax

```
APEX_WEB_SERVICE.MAKE_REST_REQUEST(
    p_url               IN VARCHAR2,
    p_http_method       IN VARCHAR2,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_scheme            IN VARCHAR2 default 'Basic',
    p_proxy_override    IN VARCHAR2 default null,
    p_transfer_timeout  IN NUMBER   default 180,
    p_body              IN CLOB default empty_clob(),
    p_body_blob         IN BLOB default empty_blob(),
    p_parm_name         IN apex_application_global.VC_ARR2 default empty_vc_arr,
    p_parm_value        IN apex_application_global.VC_ARR2 default empty_vc_arr,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null )
RETURN CLOB;
```

Parameters

Table 29-5 MAKE_REST_REQUEST Function Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_http_method	The HTTP method to use, PUT, POST, GET, HEAD, or DELETE.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CRED if supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_body	The HTTP payload to be sent as CLOB.
p_body_blob	The HTTP payload to be sent as binary BLOB. For example, posting a file.
p_parm_name	The name of the parameters to be used in name/value pairs.
p_parm_value	The value of the parameters to be used in name/value pairs.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.

Example

The following example calls a RESTful style Web service using the make_rest_request function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared CLOB.

```
declare
    l_clob     CLOB;
BEGIN

    l_clob := apex_web_service.make_rest_request(
        p_url => 'http://us.music.yahooapis.com/ video/v1/list/published/popular',
        p_http_method => 'GET',
        p_parm_name => apex_util.string_to_table('appid:format'),
        p_parm_value => apex_util.string_to_table('xyz:xml'));

END;
```

29.11 MAKE_REST_REQUEST_B Function

Use this function to invoke a RESTful style Web service supplying either name value pairs, a character based payload or a binary payload and returning the response in a BLOB.

Syntax

```
APEX_WEB_SERVICE.MAKE_REST_REQUEST_B(
    p_url               IN VARCHAR2,
    p_http_method       IN VARCHAR2,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_scheme            IN VARCHAR2 default 'Basic',
    p_proxy_override    IN VARCHAR2 default null,
    p_transfer_timeout  IN NUMBER   default 180,
    p_body              IN CLOB default empty_clob(),
    p_body_blob         IN BLOB default empty_blob(),
    p_parm_name         IN apex_application_global.VC_ARR2 default empty_vc_arr,
    p_parm_value        IN apex_application_global.VC_ARR2 default empty_vc_arr,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null )
RETURN BLOB;
```

Parameters

Table 29-6 MAKE_REST_REQUEST_B Function Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_http_method	The HTTP method to use, PUT, POST, GET, HEAD, or DELETE.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CREDif supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_body	The HTTP payload to be sent as CLOB.

Table 29-6 (Cont.) MAKE_REST_REQUEST_B Function Parameters

Parameter	Description
p_body_blob	The HTTP payload to be sent as binary BLOB. For example, posting a file.
p_parm_name	The name of the parameters to be used in name/value pairs.
p_parm_value	The value of the parameters to be used in name/value pairs.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.

Example

The following example calls a RESTful style Web service using the `make_rest_request` function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared BLOB.

```
declare
    l_blob     BLOB;
BEGIN

    l_blob := apex_web_service.make_rest_request_b(
        p_url => 'http://us.music.yahooapis.com/ video/v1/list/published/popular',
        p_http_method => 'GET',
        p_parm_name => apex_util.string_to_table('appid:format'),
        p_parm_value => apex_util.string_to_table('xyz:xml'));

END;
```

29.12 OAUTH_AUTHENTICATE Function

This function performs OAUTH autentication and requests an OAuth access token. The token and its expiry date are stored in the global variable `g_oauth_token`.

```
type oauth_token is record(
     token      varchar2(255),
     expires    date );
```

Note:

Currently only the Client Credentials flow is supported.

Syntax

```
APEX_WEB_SERVICE.OAUTH_AUTHENTICATE(
    p_token_url         in varchar2,
    p_client_id         in varchar2,
    p_client_secret     in varchar2,
    p_flow_type         in varchar2 default OAUTH_CLIENT_CRED,
    p_proxy_override    in varchar2 default null,
    p_transfer_timeout  in number   default 180,
```

```
    p_wallet_path       in varchar2 default null,
    p_wallet_pwd        in varchar2 default null );
```

Parameters

Table 29-7 OAUTH_AUTHENTICATE Function Parameters

Parameter	Description
p_token_url	The url endpoint of the OAuth token service.
p_client_id	OAuth Client ID to use for authentication.
p_client_secret	OAuth Client Secret to use for authentication.
p_flow_type	OAuth flow type - only OAUTH_CLIENT_CRED is supported at this time.
p_proxy_override	The proxy to use for the request.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The filesystem path to a wallet if request is https. For example, file:/usr/home/oracle/WALLETS.
p_wallet_pwd	The password to access the wallet.

Example

```
begin
    apex_web_service.oauth_authenticate(
        p_token_url     => '[URL to ORDS OAuth troken service: http(s)://{host}:
{port}/ords/.../oauth/token]',
        p_client_id     => '[client-id]',
        p_client_secret => '[client-secret]');
end;
```

29.13 OAUTH_GET_LAST_TOKEN Function

This function returns the OAuth access token received in the last OAUTH_AUTHENTICATE call. Returns NULL when the token is already expired or OAUTH_AUTHENTICATE has not been called.

Returns

The OAuth access token from the last OAUTH_AUTHENTICATE call; NULL when the token is expired.

Syntax

```
function oauth_get_last_token return varchar2;
```

Example

```
 select apex_web_service.oauth_get_last_token from dual;
```

29.14 OAUTH_SET_TOKEN Function

This function sets the OAuth Access token to be used in subsequence MAKE_REST_REQUEST calls. This procedure can be used to set a token which has been obtained by other means than with OAUTH_AUTHENTICATE (for instance, custom code).

Parameters

Table 29-8 OAUTH_SET_TOKEN Function Parameters

Parameter	Description
p_token	The OAuth access token to be used by MAKE_REST_REQUEST calls.
p_expires	Optional: The token expiry date; NULL means: No expiration date.

Example

```
begin
    apex_web_service.oauth_set_token(
        p_token =>   '{oauth access token}'
    );
end;
```

29.15 PARSE_RESPONSE Function

Use this function to parse the response from a Web service that is stored in a collection and return the result as a VARCHAR2 type.

Syntax

```
APEX_WEB_SERVICE.PARSE_RESPONSE (
    p_collection_name   IN VARCHAR2,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

Parameters

Table 29-9 PARSE_RESPONSE Function Parameters

Parameter	Description
p_collection_name	The name of the collection where the Web service response is stored.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared VARCHAR2 variable.

```
declare
    l_response_msg  VARCHAR2(4000);
BEGIN
    l_response_msg := apex_web_service.parse_response(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath =>
'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/
idc:statusMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');
END;
```

29.16 PARSE_RESPONSE_CLOB Function

Use this function to parse the response from a Web service that is stored in a collection and return the result as a CLOB type.

Syntax

```
APEX_WEB_SERVICE.PARSE_RESPONSE_CLOB (
    p_collection_name   IN VARCHAR2,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN CLOB;
```

Parameters

Table 29-10 PARSE_RESPONSE _CLOB Function Parameters

Parameter	Description
p_collection_name	The name of the collection where the Web service response is stored.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared CLOB variable.

```
declare
    l_response_msg  CLOB;
BEGIN
    l_response_msg := apex_web_service.parse_response_clob(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath=>
'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/
idc:statusMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');
END;
```

29.17 PARSE_XML Function

Use this function to parse the response from a Web service returned as an XMLTYPE and return the value requested as a VARCHAR2.

Syntax

```
APEX_WEB_SERVICE.PARSE_XML (
    p_xml               IN XMLTYPE,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

Parameters

Table 29-11 PARSE_XML Function Parameters

Parameter	Description
p_xml	The XML document as an XMLTYPE to parse.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example uses the `make_request` function to call a Web service and store the results in a local XMLTYPE variable. The `parse_xml` function is then used to pull out a specific node of the XML document stored in the `XMLTYPE` and stores it in a locally declared VARCHAR2 variable.

```
declare
    l_envelope CLOB;
    l_xml XMLTYPE;
    l_movie VARCHAR2(4000);
BEGIN
    l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <tns:GetTheatersAndMovies>
         <tns:zipCode>43221</tns:zipCode>
         <tns:radius>5</tns:radius>
      </tns:GetTheatersAndMovies>
   </soap:Body>
</soap:Envelope>';

   l_xml := apex_web_service.make_request(
     p_url => ' http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/
moviefunctions.asmx',
     p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
     p_envelope => l_envelope );

   l_movie := apex_web_service.parse_xml(
     p_xml => l_xml,
     p_xpath => ' //GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/Theater/
```

```
Movies/Movie/Name[1]',
     p_ns => ' xmlns="http://www.ignyte.com/whatsshowing"' );

END;
```

29.18 PARSE_XML_CLOB Function

Use this function to parse the response from a Web service returned as an XMLTYPE and return the value requested as a CLOB.

Syntax

```
APEX_WEB_SERVICE.PARSE_XML_CLOB (
    p_xml               IN XMLTYPE,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

Parameters

Table 29-12 PARSE_XML_CLOB Function Parameters

Parameter	Description
p_xml	The XML document as an XMLTYPE to parse.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example uses the `make_request` function to call a Web service and store the results in a local `XMLTYPE` variable. The `parse_xml` function is then used to pull out a specific node of the XML document stored in the `XMLTYPE` and stores it in a locally declared `VARCHAR2` variable.

```
declare
    l_envelope CLOB;
    l_xml XMLTYPE;
    l_movie CLOB;
BEGIN
    l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <tns:GetTheatersAndMovies>
         <tns:zipCode>43221</tns:zipCode>
         <tns:radius>5</tns:radius>
      </tns:GetTheatersAndMovies>
   </soap:Body>
</soap:Envelope>';

   l_xml := apex_web_service.make_request(
     p_url => ' http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/
moviefunctions.asmx',
     p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
     p_envelope => l_envelope );
```

```
   l_movie := apex_web_service.parse_xml_clob(
     p_xml => l_xml,
     p_xpath => ' //GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/Theater/
Movies/Movie/Name[1]',
     p_ns => ' xmlns="http://www.ignyte.com/whatsshowing"' );

END;
```

30

APEX_ZIP

This package manages the zipping and unzipping of files.

30.1 Data Types

The data types used by the APEX_ZIP package are described in this section.

t_files

```
type t_files is table of varchar2(32767) index by binary_integer;
```

30.2 ADD_FILE Procedure

This procedure adds a single file to a zip file.

Note:

After all files are added, you must call APEX_ZIP.FINISH.

Syntax

```
APEX_ZIP.ADD_FILE (
    p_zipped_blob IN OUT NOCOPY BLOB,
    p_file_name   IN VARCHAR2,
    p_content     IN BLOB );
```

Parameters

Table 30-1 ADD_FILE Procedure Parameters

Parameter	Description
p_zipped_blob	BLOB containing the zip file.
p_file_name	File name, including path, of the file to be added to the zip file.
p_content	BLOB containing the file.

Example

This example reads multiple files from a table and puts them into a single zip file.

```
declare
    l_zip_file blob;
begin
    for l_file in ( select file_name,
                           file_content
                      from my_files )
    loop
        apex_zip.add_file (
            p_zipped_blob => l_zip_file,
            p_file_name   => l_file.file_name,
            p_content     => l_file.file_content );
    end loop;

    apex_zip.finish (
        p_zipped_blob => l_zip_file );

end;
```

30.3 FINISH Procedure

This procedure completes the creation of a zip file after adding files with APEX_ZIP.ADD_FILE.

Syntax

```
APEX_ZIP.FINISH (
    p_zipped_blob IN OUT NOCOPY BLOB );
```

Parameters

Table 30-2 FINISH Procedure Parameters

Parameter	Description
p_zipped_blob	BLOB containing the zip file.

Example

See "ADD_FILE Procedure (page 30-1)" for an example.

30.4 GET_FILE_CONTENT Function

This function returns the BLOB of a file contained in a provided zip file.

Syntax

```
APEX_ZIP.GET_FILE_CONTENT (
    p_zipped_blob IN BLOB,
    p_file_name   IN VARCHAR2,
    p_encoding    IN VARCHAR2 DEFAULT NULL )
RETURN BLOB;
```

Parameters

Table 30-3 GET_FILE_CONTENT Function Parameters

Parameter	Description
p_zipped_blob	This is the BLOB containing the zip file.
p_file_name	File name, including path, of a file located in the zip file.
p_encoding	Encoding used to zip the file.

Returns

Table 30-4 GET_FILE_CONTENT Function Returns

Return	Description
BLOB	BLOB containing the zip file.

Example

See "GET_FILES Function (page 30-3)" for an example.

30.5 GET_FILES Function

This function returns an array of file names, including the path, of a provided zip file that contains a BLOB.

Syntax

```
APEX_ZIP.GET_FILES (
    p_zipped_blob IN BLOB,
    p_only_files  IN BOOLEAN DEFAULT TRUE,
    p_encoding    IN VARCHAR2 DEFAULT NULL )
RETURN t_files;
```

Parameters

Table 30-5 GET_FILES Function Parameters

Parameter	Description
p_zipped_blob	This is the zip file containing the BLOB.
p_only_files	If set to TRUE, empty directory entries are not included in the returned array. Otherwise, set to FALSE to include empty directory entries.
p_encoding	This is the encoding used to zip the file.

Returns

Table 30-6 GET_FILES Function Returns

Return	Description
t_files	A table of file names and path. See "Data Types (page 30-1)" for more details.

Example

This example demonstrates reading a zip file from a table, extracting it and storing all files of the zip file into `my_files`.

```
declare
    l_zip_file      blob;
    l_unzipped_file blob;
    l_files         apex_zip.t_files;
begin
    select file_content
        into l_zip_file
        from my_zip_files
    where file_name = 'my_file.zip';

    l_files := apex_zip.get_files (
            p_zipped_blob => l_zip_file );

    for i in 1 .. l_files.count loop
        l_unzipped_file := apex_zip.get_file_content (
            p_zipped_blob => l_zip_file,
            p_file_name   => l_files(i) );

        insert into my_files ( file_name, file_content )
        values ( l_files(i), l_unzipped_file );
    end loop;
end;
```

31

JavaScript APIs

This section describes JavaScript functions and objects included with Oracle Application Express and available on every page. You can use these functions and objects to provide client-side functionality, such as showing and hiding page elements, or making Ajax (Asynchronous JavaScript and XML) requests.

31.1 apex namespace

The `apex namespace` contains all the apex sub namespaces such as debug, navigation, and server, and a few common apex functions and properties.

31.1.1 About apex.confirm Function

The `apex.confirm` function displays a confirmation and depending on the user's choice either submits the page, or cancels a page submit. This function has 2 signatures.

See Also:

- "apex.confirm Signature 1 (page 31-2)"
- "apex.confirm Signature 2 (page 31-2)"

31.1.2 apex.confirm Signature 1

This function is an alias for `apex.page.confirm`.

See Also:

"apex.page.confirm Signature 1 (page 31-50)"

31.1.3 apex.confirm Signature 2

This function is an alias for `apex.page.confirm`.

See Also:

"apex.page.confirm Signature 2 (page 31-50)"

31.1.4 apex.gPageContext$ Property

Application Express variable that stores the current page context. The current page context is different depending on whether the page is a Desktop, or jQuery Mobile page. For Desktop, this is set to the document level. For jQuery Mobile, where pages are actually represented as DIV elements in the Browser DOM and multiple page DIVs can be loaded in the Browser DOM at one time, this is set to the DIV element representing the current page. This is used to set the context for your jQuery selectors, to ensure that the selector is executing within the context of the correct page.

Example

```
jQuery( ".my_class", apex.gPageContext$ );
```

This selects all elements with a CSS class of `my_class`, in the context of the current page.

31.1.5 apex.jQuery Property

Application Express property that holds the jQuery object that Application Express uses. Ideally there is just one copy of jQuery on a page but it is possible to have multiple copies and even different versions of jQuery on a page. This is sometimes necessary when using third party plugins that only work with an older version of jQuery. Use this property in place of global variables $ or jQuery to ensure you are using the same jQuery library that Application Express is using.

Example

The following function creates a local variable $ as a convenient way to reference jQuery while ensuring that it is using the same jQuery that APEX uses.

```
function myFunction() {
    var $ = apex.jQuery;
    // use $ to access jQuery functionality
}
```

31.1.6 About apex.submit Function

The `apex.submit` function submits the current page. This function has 2 signatures.

See Also:

- apex.submit Signature 1 (page 31-3)
- apex.submit Signature 2 (page 31-3)

31.1.7 apex.submit Signature 1

This function is an alias for `apex.page.submit`.

See Also:

"apex.page.submit Signature 2 (page 31-52)"

31.1.8 apex.submit Signature 2

This function is an alias for `apex.page.submit`.

See Also:

"apex.page.submit Signature 1 (page 31-51)"

31.2 apex.da namespace

This namespace holds all Dynamic Action functions in Oracle Application Express.

apex.da.resume (page 31-4)

31.2.1 apex.da.resume

This function resumes execution of a Dynamic Action. Execution of a Dynamic Action can be paused, if the action's `Wait for Result` attribute is checked. The `Wait for Result` is a Dynamic Action plug-in standard attribute designed for use with Ajax based Dynamic Actions. If a plug-in exposes this attribute, it needs to resume execution by calling this function in the relevant place in the plug-in JavaScript code, otherwise, your action breaks execution of Dynamic Actions.

Parameters

Table 31-1 apex.da.resume

Name	Type	Description
`pCallback`	Function	This is a required parameter that references a callback function available from the `this.resumeCallback` property.
`pErrorOccurred`	Boolean	This is a required parameter that indicates to the framework whether an error has occurred. If an error has occurred and the action's Stop Execution on Error attribute is checked, execution of the Dynamic Action is stopped.

Example 1

Resume execution of the actions indicating that no error has occurred, for example from a success callback of an Ajax based action.

```
apex.da.resume( lResumeCallback, false );
```

Example 2

Resume execution of the actions indicating that an error has occurred, for example from an error callback of an Ajax based action. If the action's

`Stop Execution on Error` attribute is checked, execution of the dynamic action is stopped.

```
apex.da.resume( lResumeCallback, true );
```

31.3 apex.debug namespace

This namespace stores all debug functions of Oracle Application Express.

31.3.1 Log Level Constants

LOG_LEVEL

```
apex.debug.LOG_LEVEL = {
        OFF: 0,
        ERROR: 1,
        WARN: 2,
        INFO: 4,
        APP_TRACE: 6,
        ENGINE_TRACE: 9
    };
```

Table 31-2 LOG_LEVEL Descriptions

Value	Description
`OFF: 0`	Logging is off.
`ERROR: 1`	Error logging level
`WARN: 2`	Warning logging level.
`INFO: 4`	Information logging level.
`APP_TRACE: 6`	Application tracing logging level.
`ENGINE_TRACE: 9`	Engine tracing logging level.

31.3.2 apex.debug.error

Log an error message. The error function always writes the error regardless of the log level from the server or set with apex.debug.setLevel. Messages are written using the browsers built-in console logging if available. If supported `console.trace` is called. Older browsers may not support the console object or all of its features.

Parameters

Table 31-3 Parameters for debug.error

Name	Type	Description
...*	any	Any number of parameters logged to the console.

Example 1

This example writes the message "Update Failed" to the console.

```
apex.debug.error("Update Failed");
```

Example 2

This example writes an exception message to the console.

```
apex.debug.error("Exception: ", ex);
```

31.3.3 apex.debug.getLevel

Method that returns the debug log level. The debug log level is synchronized with hidden item "#pdebug".

Parameters

None

Returns

Returns logging level as an integer 1 to 9 or 0 to indicate debug logging is turned off.

Example

This example retrieves the logging level, prepends `"Level"` and logs to the console.

```
apex.debug.log("Level=", apex.debug.getLevel());
```

See Also:

"Log Level Constants (page 31-5)" for return value meanings.

31.3.4 apex.debug.info

Log an informational message. Similar to`apex.debug.message` with the level set to INFO.

Parameters

Table 31-4 Parameters for debug.info

Name	Type	Description
...*	any	Any number of parameters logged to the console.

Example

This example prints an informational message to the console if the log level is INFO or greater.

```
apex.debug.info("Command successful");
```

31.3.5 apex.debug.log

Log a message. Similar to `apex.debug.message` with the level set to the highest level.

Parameters

Table 31-5 Parameters for debug.log

Name	Type	Description
...*	any	Any number of parameters logged to the console.

Example

This example gets the logging level and writes it to the console, regardless of the current logging level.

```
apex.debug.log("Level=", apex.debug.getLevel());
```

31.3.6 apex.debug.message

Log a message at the given debug log level. The log level set from the server or with `apex.debug.setLevel` controls if the message is actually written. If the set log level is >= pLevel then the message is written. Messages are written using the browsers built-in console logging if available. Older browsers may not support the console object or all of its features.

Parameters

Table 31-6 Parameters for debug.message

Name	Type	Description
`pLevel`	Number	A number from 1 to 9 where level 1 is most important and level 9 is least important. Can be one of the `LOG_LEVEL` constants. Any other value such as 0 will turn off debug logging.
...*	any	Any number of parameters logged to the console.

Example

This example writes the message "Testing" to the console if the logging level is greater than or equal to 7.

```
apex.debug.message(7,"Testing"));
```

31.3.7 apex.debug.setLevel

Method that sets the debug log level. Log messages at or below the specified level are written to the console log. It is rarely necessary to call this function because the debug log level is synchronized with the hidden item `#pdebug` that comes from the server.

Parameters

Table 31-7 Parameters for debug.setlevel

Name	Type	Description
pLevel	Number	A number from 1 to 9 where level 1 is most important and level 9 is least important. Can be one of the LOG_LEVEL constants. Any other value such as 0 will turn off debug logging.

Example

This example sets the logging level to application tracing.

```
apex.debug.setLevel(apex.debug.LOG_LEVEL.APP_TRACE));
```

31.3.8 apex.debug.trace

Log a trace message. Similar to `apex.debug.message` with the level set to APP_TRACE.

Parameters

Table 31-8 Parameters for debug.trace

Name	Type	Description
...*	any	Any number of parameters logged to the console.

Example

This example writes a log message to the console if the debug log level is APP_TRACE or greater.

```
apex.debug.trace("Got click event: ", event);
```

31.3.9 apex.debug.warn

Log a warning message. Similar to `apex.debug.message` with the level set to WARN.

Parameters

Table 31-9 Parameters for debug.warn

Name	Type	Description
...*	any	Any number of parameters logged to the console.

Example

This example writes a warning message to the console if the debug log level is WARN or greater.

```
apex.debug.warn("Empty string ignored");
```

31.4 apex.event namespace

The `apex.event` namespace stores all event related functions of Oracle Application Express.

31.4.1 apex.event.trigger

Given a jQuery selector, jQuery object or DOM Node the specified pEvent is triggered. pEvent can be a browser event like "click" or "change" but also a custom event like "slidechange". This function should only be used to trigger events that are handled by the dynamic action framework. Otherwise, custom events registered by plug-ins installed in your application or any event that is already exposed in dynamic actions can be compromised.

Parameters

Table 31-10 apex.event.trigger

Name	Type	Description
`pSelector`	jQuery selector | jQuery object | DOM Node	Selector for which the event will be triggered.
`pEvent`	String	The name of the event.
`pData`	Object	Optional array of addition arguments to pass to the event.

Returns

Returns `true` if the event is canceled.

`Boolean`

31.5 apex.item

The `apex.item` API provides a single interface for item related functionality of Application Express. This API returns an Application Express item object, which can then be used to access item related functions and properties. Item objects can apply to either page items or column items. Page items are items on the page backed by session state in any region. Column items are created by region types such as Interactive Grid that support editable columns. The state of a column item, including its value, changes according to the editing context of the region.

31.5.1 apex.item

This API returns an Application Express item object, which can then be used to access item related functions and properties.

Plug-in developers can override much of the behavior defined in the `apex.item` namespace, by calling `apex.item.create` with their overrides.

Parameters

Table 31-11 Parameters for pNd

Name	Type	Description
`pNd`	DOM Node | String	Application Express item name or DOM node. This parameter can refer to either a page item or column item.

Returns

Returns the Application Express item object, which is used to access item specific functions. For example `getValue`, `setValue`, and so on.

Examples

This will not be used by itself, rather it is used to access item specific functions and properties, as documented in the following APIs

See Also:

"apex.item.create (page 31-19)"

31.5.2 addValue

Adds a value to an Application Express item that supports multiple values.

Parameters

Table 31-12 Parameters for addValue

Name	Type	Description
pValue	String	The value to be added.

Examples

In this example, the page item called P1_ITEM will have the value 100 added to the values currently selected.

```
apex.item( "P1_ITEM" ).addValue('100') ;
```

31.5.3 disable

Disables the Application Express item value, taking into account the item type, making it unavailable for edit.

Parameters

None.

Examples

In this example, the page item called P1_ITEM will be disabled and unavailable for edit.

```
apex.item( "P1_ITEM" ).disable() ;
```

31.5.4 displayValueFor

Returns the display value corresponding to the value given by pValue for the Application Express item. If the item type does not have a display value distinct from the value then pValue is returned; meaning that the value is the display value. For item types that have a display value but don't have access to all possible values and display values then this function only works when pValue is the current value of the item. For the native items this only applies to item type Popup LOV, with the attribute Input Field" = "Not Enterable, Show Display Value and Store Return Value. For item types such as select lists that have access to all their values, if pValue is not a valid value then pValue is returned.

Parameters

Table 31-13 Parameters displayValueFor

Name	Type	Description
pValue	String	A valid value for this item.

Returns

The string display value corresponding to the given `pValue` as described above.

Example

This example gets a display value from a select list item called `P1_ITEM` and displays it in an alert.

```
alert( "The correct answer is: " +
apex.item( "P1_ITEM" ).displayValueFor( "APPLES" ) );
```

31.5.5 enable

Enables the Application Express item value, taking into account the item type, making it available for edit.

Parameters

None.

Examples

In this example, the page item called `P1_ITEM` will be enabled and available for edit.

```
apex.item( "P1_ITEM" ).enable() ;
```

31.5.6 getValidity

Return a ValidityState object as defined by the HTML5 constraint validation API for the Application Express item. If a plug-in item implements its own validation then the object may not contain all the fields defined by HTML5. At a minimum it must have the *valid* property. If the item doesn't support HTML5 validation then it is assumed to be valid.

This function does not actually validate the item value. For many item types the browser can do the validation automatically if you add HTML5 constraint attributes such as pattern. Validation can be done using the HTML5 constraint validation API.

Developers rarely have a need to call this function. It is used internally by the client side validation feature. Item Plug-in developers should ensure this function works with their plug-in.

Parameters

None.

Returns

A ValidityState object as described above.

Example

The following example displays a message in an alert dialog if the item called `P1_ITEM` is not valid.

```
var item = apex.item( "P1_ITEM" );
if ( !item.getValidity().valid ) {
    alert( "Error: " + item.getValidationMessage() );
}
```

31.5.7 getValidationMessage

Return a validation message if the Application Express item is not valid and empty string otherwise.

The message comes from the element's `validationMessage` property. An APEX extension allows specifying a custom message, which overrides the element's `validationMessage`, by adding a custom attribute named *data-valid-message*. If the item has this attribute then its value is returned if the item is not valid. As the name implies the text of the message should describe what is expected of valid input rather than what went wrong.

Parameters

None.

Returns

A validation message if the item is not valid and empty string otherwise.

Example

See the example for `getValidity` for an example of this function.

31.5.8 getValue

Returns the current value of an Application Express item on a page, taking into account the current item type. This does not return the item's current value from session state (although that could be the same), rather it will return the value as it is on the current page.

There are 2 related functions to .getValue(). $v(pNd) which returns an item's value, but in the format it will be posted. This will either be a single value, or if the item supports multiple values, will be a ':' colon separated list of values. There is also the $v2(pNd) function, which is just a shortcut to .getValue() and returns either a single value, or array of values.

Parameters

None.

Returns

Returns either a single string value or array of string values if the item supports multiple values (for example the 'Select List' with attribute 'Allow Multi Selection' set to ' Yes' or 'Shuttle' native item types).

Examples

In this example, the current value of the page item called `P1_ITEM` will be shown in an alert.

```
alert( "P1_ITEM value = " +  apex.item( "P1_ITEM" ).getValue()  );
```

31.5.9 hide

Hides the Application Express item value, taking into account the item type. When using the .hide() function, it is important to understand the following:

- If the item being hidden is rendered on a page using table layout (meaning the page references a page template with Grid Layout Type set to 'HTML Table'), and the call to hide has specified to hide the entire table row (pHideRow = true), then it is assumed that everything pertaining to the item is contained in that row, and the entire row will be hidden.
- If the item being hidden is rendered on a page using table layout, and the call to hide has specified not to hide the entire table row (pHideRow = false, or not passed), then the function will attempt to hide the item's label, where the FOR attribute matches the ID of the item.
- If the item being hidden is rendered on a page using grid layout (meaning the page references a page template with Grid Layout Type set to either 'Fixed Number of Columns', or 'Variable Number of Columns'), and the item references a Label template that includes a Field Container element with a known ID (so where the Field Container > Before Label and Item attribute includes an HTML element with id="#CURRENT_ITEM_CONTAINER_ID#"), then it is assumed that everything pertaining to the item is contained in the Field Container, and this will be hidden.
- If the item is a column item then just the column value is hidden. The exact behavior depends on the type of region. For example in Interactive Grid just the cell content is hidden not the whole column.

Parameters

Table 31-14 Parameters for hide

Name	Type	Description
`pHideRow`	Boolean	This parameter is optional.The default value is `false`. If `true`, hides the nearest containing table row (TR). This parameter is not supported for column items. Its behavior is undefined. Only applicable when item is on a page using table layout (meaning the page references a page template with Grid Layout Type set to 'HTML Table').

Examples

In this example, the page item called `P1_ITEM` will be hidden. If `P1_ITEM` is on a page using grid layout and the item references a Label template that includes a Field Container element with a known ID (as detailed above), then that container element will be hidden. Otherwise just the item and its corresponding label will be hidden.

```
apex.item( "P1_ITEM" ).hide();
```

In this example, the page item called P1_ITEM's nearest containing table row (TR) will be hidden (as pHideRow = true). Hiding the entire table row should only be used on a page using table layout. If `P1_ITEM` is on a page using grid layout, then passing

`pHideRow = true` will not work and could result in adverse consequence for the page layout, where an incorrect table row is wrongly hidden.

```
apex.item( "P1_ITEM" ).hide(true);
```

31.5.10 isChanged

Return `true` if the current value of the Application Express item has changed and `false` otherwise. Developers rarely have a need to call this function. It is used internally by the Warn on Unsaved Changes feature. Item Plug-in developers should ensure this function works so that the Warn on Unsaved Changes feature can support their plug-in.

Parameters

None.

Returns

`true` if the current value has changed and `false` otherwise.

Examples

The following example determines if the value of item `P1_ITEM` has been changed.

```
If ( apex.item( "P1_ITEM" ).isChanged() ) {
    // do something
}
```

31.5.11 isDisabled

Returns the disabled state of an item

Parameters

None.

Returns

`true` if the Application Express item is disabled and`false` otherwise.

Example

This example gets the value of an item but only if it is not disabled.

```
var value = null;
if ( !apex.item( "P1_ITEM" ).isDisabled() ) {
    value = apex.item( "P1_ITEM" ).getValue(); }
```

31.5.12 isEmpty

Returns `true` or `false` if an Application Express item is empty and considers any item value consisting of only whitespace including space, tab, or form-feed, as empty. This also respects if the item type uses a List of Values, and a 'Null Return Value' has defined in the List of Values. In that case, the 'Null Return Value' is used to assert if the item is empty.

Parameters

None.

Returns

`true` if the Application Express item is empty and `false` otherwise.

Examples

In this example, the call to `.isEmpty()` determines if the page item called `P1_ITEM` is empty, and if so displays an alert.

```
if( apex.item( "P1_ITEM" ).isEmpty()  ) {
  alert( "P1_ITEM empty!" );
}
```

31.5.13 Item Object Properties

An item object returned from `apex.item` has the properties given in the following table.

Parameters

Table 31-15 Item Object Properties

Name	Description
`id`	The id property of an Application Express item provides the id of the item DOM element.
`node`	The node property of an Application Express item provides the DOM element of the item. If the item is invalid then the value is `false`.

Example

The following code checks if the Application Express item `P1_OPTIONAL_ITEM` exists before setting its value. Use code similar to this if there is a possibility of the item not existing.

```
var item = apex.item( "P1_OPTIONAL_ITEM" );
if ( item.node ) {
    item.setValue( newValue );
}
```

31.5.14 setFocus

Places user focus on the Application Express item, taking into account how specific items are designed to receive focus.

Parameters

None.

Examples

In this example, user focus is set to the page item called `P1_ITEM`.

```
apex.item( "P1_ITEM" ).setFocus();
```

31.5.15 setStyle

Sets a style for the Application Express item, taking into account how specific items are designed to be styled.

Note:

Using setStyle is not a best practice. It is better to add or remove CSS classes and use CSS rules to control the style of items. Also keep in mind that the exact markup of native and plug-in items can change from one release to the next.

Parameters

Table 31-16 Parameters for setStyle

Name	Type	Description
`pPropertyName`	String	The CSS property name that will be set.
`pPropertyValue`	String	The value used to set the CSS property.

Example

In this example, the CSS property `color` will be set to `red` for the page item called `P1_ITEM`.

```
apex.item( "P1_ITEM" ).setStyle( "color", "red" );
```

31.5.16 setValue

Sets the Application Express item value, taking into account the item type. This function sets the current value of an Application Express item on the page, not the item's current value in session state. It also allows for the caller to suppress the 'change' event for the item being set, if desired.

See the `$s( pNd, pValue, pDisplayValue, pSuppressChangeEvent )` function for a shortcut to `.setValue()`.

Parameters

Table 31-17 Parameters for setValue

Name	Type	Description
`pValue`	String | Array	The value to be set. For items that support multiple values (for example a 'Shuttle'), an array of string values can be passed to set multiple values at once.

Table 31-17 (Cont.) Parameters for setValue

Name	Type	Description
pDisplayValue	String	Optional parameter used to set the page item's display value, in the case where the return value is different. For example for the item type Popup LOV, with the attribute Input Field = Not Enterable, Show Display Value and Store Return Value, this value sets the Input Field. The value of pValue is then used to set the item's hidden return field.
pSuppressChange Event	Boolean	This parameter is optional. The default if not specified is false. Pass true to prevent thechange event from being triggered, for the item being set.

Examples

In this example, the value of the page item called P1_ITEM will be set to 10. As pSuppressChangeEvent has not been passed, the default behavior of the change event triggering for P1_ITEM will occur.

```
apex.item( "P1_ITEM" ).setValue( "10" );
```

In this example P1_ITEM is a Popup LOV page item with the attribute Input Field = Not Enterable, Show Display Value and Store Return Value, set to Input Field. The display value of P1_ITEM will be set to SALES and the hidden return value will be set to 10. As true has been passed for the pSuppressChangeEvent parameter, the 'change' event will not trigger for the P1_ITEM item.

```
apex.item( "P1_ITEM" ).setValue( "10", "SALES", true );
```

31.5.17 show

Shows the Application Express item value, taking into account the item type. When using the .show() function, it is important to understand the following:

- If the item being shown is rendered on a page using table layout (meaning the page references a page template with Grid Layout Type set to 'HTML Table'), and the call to show has specified to show the entire table row (pShowRow = true), then it is assumed that everything pertaining to the item is contained in that row, and the entire row will be shown.
- If the item being shown is rendered on a page using table layout, and the call to show has specified not to show the entire table row (pShowRow = false, or not passed), then the function will attempt to show the item's label, where the FOR attribute matches the ID of the item.
- If the item being shown is rendered on a page using grid layout (meaning the page references a page template with Grid Layout Type set to either 'Fixed Number of Columns', or 'Variable Number of Columns'), and the item references a Label template that includes a Field Container element with a known ID (so where the Field Container > Before Label and Item attribute includes an HTML element with id="#CURRENT_ITEM_CONTAINER_ID#"), then it is assumed that everything pertaining to the item is contained in the Field Container, and this will be shown.

- If the item is a column item then just the column value is shown. The exact behavior depends on the type of region. For example in Interactive Grid just the cell content is shown not the whole column.

Parameters

Table 31-18 Parameters for show

Name	Type	Description
pShowRow	Boolean	This parameter is optional. The default if not specified is `false`. If `true`, shows the nearest containing table row (TR). This parameter is not supported for column items. Its behavior is undefined. Only applicable when item is on a page using table layout (meaning the page references a page template with Grid Layout Type set to 'HTML Table').

Examples

In this example, the page item called `P1_ITEM` will be shown. If `P1_ITEM` is on a page using grid layout and the item references a Label template that includes a Field Container element with a known ID (as detailed above), then that container element will be shown. Otherwise just the item and its corresponding label will be shown.

```
apex.item( "P1_ITEM" ).show();
```

In this example, the page item called P1_ITEM's nearest containing table row (TR) will be shown (as pShowRow = true). Showing the entire table row should only be used on a page using table layout. If `P1_ITEM` is on a page using grid layout, then passing `pShowRow = true` will not work and could result in adverse consequence for the page layout, where an incorrect table row is wrongly shown.

```
apex.item( "P1_ITEM" ).show(true);
```

31.5.18 apex.item.create

This function is only for item plug-in developers. It provides a plug-in specific implementation for the item. This is necessary to seamlessly integrate a plug-in item type with the built-in item related client-side functionality of Application Express.

Parameters

Table 31-19 apex.item.create Parameters

Name	Type	Description
pName	DOM Node| String	Application Express page item name or DOM node. This parameter can refer to either a page item or column item.

Table 31-19 (Cont.) apex.item.create Parameters

Name	Type	Description
`pItemImpl`	Object	An object with properties that provide any functions needed to customize the Application Express item object behavior. The object returned by `apex.item` has default implementations for each of its functions that are appropriate for many page items particularly for items that use standard form elements. For each function of `apex.item` you should check if the default handling is appropriate for your item plug-in. If it isn't you can provide your own implementation of the corresponding function through this `pItemImpl` object. The default behavior is used for any functions omitted. `pItemImpl` can contain any of the following properties: • `displayValueFor( pValue )` • `enable()` • `getPopupSelector()` • `getValidity()` • `getValue()` • `setValue( pValue, pDisplayValue )` • `disable()` • `show()` • `hide()` • `isChanged()` • `isDisabled()` • `addValue()` • `nullValue()` • `setFocusTo` • `setStyleTo` • `afterModify()` • `loadingIndicator( pLoadingIndicator$ )`

Table 31-20 Properties for the pItemImpl parameter

Name	Description
`displayValueFor( pValue )`	Specify a function that returns a string display value that corresponds to the given pValue.
`enable()`	Specify a function for enabling the item, which overrides the default page item handling. This could be useful for example where the item consists of compound elements which also need enabling, or if the item is based on a widget that already has its own enable method that you want to reuse. Ensuring the item can enable correctly means certain item related client-side functionality of Application Express still works, for example when using the Enable action of a Dynamic Actions, to enable the item. Note: Even if this function is defined, the default handling always handles the logic associated with the `.afterModify()` function, so that is outside the scope of what a plug-in developer is concerned with. See the "enable (page 31-12)," for details on how to define this function.

Table 31-20 (Cont.) Properties for the pItemImpl parameter

Name	Description
`getPopupSelector()`	Specify a function that returns a CSS selector that locates the popup used by the item. Any plug-in item type that uses a popup (a div added near the end of the document that is positioned near the input item and floating above it) needs to provide a CSS selector that locates the top level element of the popup. This allows the item type to be used in the Interactive Grid region or any other region that needs to coordinate focus with the popup. The default implementation returns null. In addition the top level popup element must be focusable (have attribute tabindex = -1). For best behavior of a popup in the Interactive Grid. The popup should: • have a way of taking focus • close on escape when it has focus • close when the element it is attached to looses focus • return focus to the element that opened the popup when it closes • manage its tab stops so they cycle in the popup or return to the element that opened the popup at the ends
`getValidity()`	Specify a function that returns a validity object. The returned object must at a minimum have the Boolean valid property. It may include any of the properties defined for the HTML5 ValidityState object. The default implementation returns the validity object of the item element if there is one otherwise it returns { valid: true }. See the"getValidity (page 31-12)" or details on how to define this function.
`getValue()`	Specify a function for getting the item's value, which overrides the default page item handling. Ensuring the item returns its value correctly means certain item related client-side functionality of Application Express still works, for example in Dynamic Actions to evaluate a When condition on the item, or when calling the JavaScript function $v to get the item's value. See "getValue (page 31-13)," for details on how to define this function.

Table 31-20 (Cont.) Properties for the pItemImpl parameter

Name	Description
`setValue( pValue, pDisplayValue )`	Specify a function for setting the item's value, which overrides the default page item handling. Ensuring the item can set its value correctly means certain item related client-side functionality of Application Express still works, for example when using the Set Value action of a Dynamic Action to set the item's value, or when calling the JavaScript function $s to set the item's value. Note: Even if this function is defined, the default handling always handles the logic associated with the `.afterModify()` function and the `pSuppressChangeEvent` parameter, so that is outside the scope of what a plug-in developer is concerned with. See the "setValue (page 31-17)," for details on how to define this function.
`disable()`	Specify a function for disabling the item, which overrides the default page item handling. This could be useful for example where the item consists of compound elements which also need disabling, or if the item is based on a widget that already has its own disable method that you want to reuse. Ensuring the item can disable correctly means certain item related client-side functionality of Application Express still works, for example when using the Disable action of a Dynamic Action to disable the item. Note: Even if this function is defined, the default handling always handles the logic associated with the `.afterModify()` function, so that is outside the scope of what a plug-in developer is concerned with. See the "disable (page 31-11)," for details on how to define this function.
`show()`	Specify a function for showing the item, which overrides the default page item handling. This is useful for example where the item consists of compound elements which also need showing, or if the item is based on a widget that already has its own show method that you want to reuse. Ensuring the item can show correctly means certain item related client-side functionality of Application Express still works, for example when using the Show action of a Dynamic Action, to show the item. See the "show (page 31-18)," for details on how to define this function.
`hide()`	Specify a function for hiding the item, which overrides the default page item handling. This could be useful for example where the item consists of compound elements which also needs hiding, or if the item is based on a widget that already has its own hide method that you want to reuse. Ensuring the item can hide correctly means certain item related client-side functionality of Application Express still works, for example when using the Hide action of a Dynamic Action, to hide the item. See the "hide (page 31-14)," for details on how this function should be defined.

Table 31-20 (Cont.) Properties for the pItemImpl parameter

Name	Description
`isChanged()`	Specify a function that returns `true` if the current value of the item has changed and `false` otherwise. This function allows the Warn on Unsaved Changes feature to work. The default implementation uses built-in functionality of HTML form elements to detect changes. If this function does not work correctly then changes to the plug-in item type value will not be detect and the user will not be warned when they leave the page. See the "isChanged (page 31-15)" for details on how this function should be defined.
`isDisabled()`	Specify a function that returns `true` if the item is disabled and `false` otherwise.
`addValue()`	Specify a function for adding a value to the item, where the item supports multiple values. Currently there is no client-side functionality of Application Express dependent on this. There is also no default page item handling. Note: Even if this function is defined, the default handling always handles the logic associated with the `.afterModify()` function, so that is outside the scope of what a plug-in developer is concerned with. See the "addValue (page 31-11)," for details on how this function should be defined.

Table 31-20 (Cont.) Properties for the pItemImpl parameter

Name	Description

`nullValue`	Specify a value that to be used to determine if the item is null. This is used when the item supports definition of a List of Values, where a developer can define a Null Return Value for the item and where the default item handling needs to know this in order to assert if the item is null or empty. This can be done by following these steps:
	
	1. From the Render function in the plug-in definition, emit the value stored in `p_item.lov_null_value` as part of the item initialization JavaScript code that fires when the page loads. For example:
	
	`/* Assumes that you have some JavaScript function called 'com_your_company_your_item' that accepts 2 parameters, the first being the name of the item and the second being an object storing properties (say pOptions) required by the item's client side code. */`
	`apex_javascript.add_onload_code (`
	`p_code => 'com_your_company_your_item('||`
	`apex_javascript.add_value(`
	
	`apex_plugin_util.page_item_names_to_jquery(p_item.name)||', {'||`
	
	`apex_javascript.add_attribute('lovNullValue', p_item.lov_null_value, false, false)||`
	`'});' );`
	
	2. Then, in the implementation of `com_your_company_your_item( pName, pOptions )` you have the value defined for the specific item's Null Return Value in the `pOptions.lovNullValue` property. This can then be used in your call to `apex.widget.initPageItem`, to set the `nullValue` property.
	
	Ensuring the `nullValue` property is set means certain item related client-side functionality of Application Express still works, for example, in Dynamic Actions to correctly evaluate an `is null` or `is not null` when condition on the item, or when calling the JavaScript function `apex.item( pNd ).isEmpty()` to determine if the item is null.
	
	See the "isEmpty (page 31-15)," for further details of this API.

Table 31-20 (Cont.) Properties for the pItemImpl parameter

Name	Description
`setFocusTo`	Specify the element to receive focus, when focus is set to the item using the `apex.item( pNd ).setFocus()` API. This can be defined as either a jQuery selector, jQuery or DOM object which identifies the DOM element, or a function that returns a jQuery object referencing the element. This can be useful when the item consists of compound elements, and you do not want focus to go to the element that has an ID matching the item name, which is the default behavior. For example, the native item type `Popup LOV` when the attribute `Input Field` is set to `Not enterable, Show Display Value and Store Return Value` renders a disabled input field as the main element with an ID matching the item name and a popup selection icon next to the input. In this case, because you do not want focus to go to the disabled input, use the `setFocusTo` item property and set that to the popup selection icon. Ensuring the item sets focus correctly means certain item related client-side functionality of Application Express still works, for example when using the `Set Focus` action of a Dynamic Action to set focus to the item, when users follow the `Go to Error` link that displays in a validation error message to go straight to the associated item, or when the item is the first item on a page and the developer has the page level attribute `Cursor Focus` set to `First item on page`. See the "setFocus (page 31-16)," for further details of this API.
`setStyleTo`	Specify the element to receive style, when style is set to the item using the `apex.item( pNd ).setStyle()` API. This can be defined as either a jQuery selector, jQuery- or DOM object which identifies the DOM element(s), or a function that returns a jQuery object referencing the element(s). This is useful when the item consists of compound elements, and you do not want style to be set to the element or just the element, that has an ID matching the item name which is the default behavior. Ensuring the item sets style correctly means certain item related client-side functionality of Application Express still works, for example when using the `Set Style` action of a Dynamic Action to add style to the item. Note: Even if this property is defined, the default handling still always handles the logic associated with the `.afterModify()` function, so that is outside the scope of what a plug-in developer is concerned with. See the `apex.item( pNd ).setStyle()` documentation, for further details of this API.
`afterModify()`	Specify a function that is called after an item is modified. This is useful, for example as some frameworks such as jQuery Mobile need to be notified if widgets are modified, for example their value has been set, or they have been disabled in order to keep both the native and enhanced controls in sync. This callback provides the hook to do so.

Table 31-20 (Cont.) Properties for the pItemImpl parameter

Name	Description
`loadingIndicator( pLoadingIndicator$ )`	Specify a function that normalizes how the item's loading indicator is displayed during a partial page refresh of the item. This function must pass the `pLoadingIndicator$` parameter as the first parameter, which contains a jQuery object with a reference to the DOM element for the loading indicator. The function then adds this loading indicator to the appropriate DOM element on the page for the item, and also returns the jQuery object reference to the loading indicator, such that the framework has a reference to it, so it can remove it once the call is complete. This is used, for example, if the item is a `Cascading LOV` and the Cascading LOV Parent Item changes, or when setting the item's value by using one of the server-side Dynamic Actions such as `Set Value - SQL Statement`.

Example

The following example shows a call to `apex.item.create( pNd, pItemImpl )` with all the available callbacks and properties passed. It is unlikely that any plug-in needs to supply every callback and property. This is just to illustrate the syntax.

```
apex.item.create( "P100_COMPANY_NAME", {
     displayValueFor: function( pValue ) {
        var lDisplayValue;
        // code to determine the display value for pValue
        return lDisplayValue;
    },
     enable: function() {
         // code that enables the item type
     },
     getPopupSelector: function() {
         return "<some CSS selector>";
    },
     getValidity: function() {
         var lValidity = { valid: true };
         if ( /* item is not valid */ ) {
             lValidity.valid = false;
         }
         return lValidity;
     },
     getValue: function() {
         var lValue;
         // code to determine lValue based on the item type.
         return lValue;
     },
     setValue: function( pValue, pDisplayValue ) {
        // code that sets pValue and pDisplayValue (if required), for the item type
     },
     disable: function() {
        // code that disables the item type
     },
     show: function() {
         // code that shows the item type
     },
     hide: function() {
```

```
        // code that hides the item type
    },
    isChanged: function() {
        var lChanged;
        // code to determine if the value has changed
        return lChanged;
    },
    addValue: function( pValue ) {
        // code that adds pValue to the values already in the item type
    },
    nullValue: "<null return value for the item>",
    setFocusTo: $( "<some jQuery selector>" ),
    setStyleTo: $( "<some jQuery selector>" ),
    afterModify: function(){
        // code to always fire after the item has been modified (value set,
enabled, etc.)
    },
    loadingIndicator: function( pLoadingIndicator$ ){
        // code to add the loading indicator in the best place for the item
        return pLoadingIndicator$;
    }
});
```

See Also:

"Implementing Plug-ins" in *Oracle Application Express App Builder User's Guide*

31.6 apex.lang namespace

This namespace is used for localization related functions of Oracle Application Express.

31.6.1 apex.lang.addMessages

Add messages for use by `getMessage` and the format functions. Can be called multiple times. Additional messages are merged. It is generally not necessary to call this function, because it is automatically called with all the application text messages that have Used in JavaScript set to Yes.

Parameters

Table 31-21 Parameters for apex.lang.addMessages

Name	Type	Description
pMessages	Object	An object whose properties are message keys and the values are localized message text.

Example

This example adds a message.

```
apex.lang.addMessages({
  APPLY_BUTTON_LABEL: "Apply"
});
```

31.6.2 apex.lang.clearMessages

Remove all messages.

Parameters

None.

Example

This example removes all messages.

```
apex.lang.clearMessages()
```

31.6.3 apex.lang.format

Same as formatMessage except the message pattern is given directly (already localized or isn't supposed to be). It is not a key. The replacement arguments are HTML escaped.

Parameters

Table 31-22 Parameters for apex.lang.format

Name	Type	Description
pPattern	String	The message pattern that contains one or more parameters %0 to %9.
...*	any	Optional replacement values one for each message parameter %0 to %9. Non string arguments are converted to strings.

Returns

The localized and formatted message text string.

Example

This example returns **Total cost: $34.00** assuming the `orderTotal` variable equals **34.00**.

```
apex.lang.format("Total cost: $%0", orderTotal);
```

See Also:

"apex.lang.formatMessage (page 31-29)"

31.6.4 apex.lang.formatMessage

Format a message. Parameters in the message %0 to %9 are replaced with the corresponding function argument. Use %% to include a single %. The replacement arguments are HTML escaped.

Parameters

Table 31-23 Parameters for apex.lang.formatMessage

Name	Type	Description
pKey	String	The key is used to lookup the localized message text as if with getMessage.
...*	any	Optional replacement values one for each message parameter %0 to %9. Non string arguments are converted to strings.

Returns

The localized and formatted message text string. If the key is not found then the key is returned.

Example

This example returns "Process 60% complete" when the PROCESS_STATUS message text is "Process %0%% complete" and the progress variable value is 60.

```
apex.lang.formatMessage("PROCESS_STATUS", progress);
```

31.6.5 apex.lang.formatMessageNoEscape

Same as formatMessage except the replacement arguments are not HTML escaped. They must be known to be safe or are used in a context that is safe. See "apex.lang.formatMessage (page 31-29)".

Parameters

Table 31-24 Parameters for apex.lang.formatMessageNoEscape

Name	Type	Description
pKey	String	The key is used to lookup the localized message text as if with getMessage.
...*	any	Optional replacement values one for each message parameter %0 to %9.

Returns

The localized and formatted message text string. If the key is not found then the key is returned.

Example

This example returns "You entered <ok>" when the CONFIRM message text is "You entered %0" and the inputValue variable value is "<ok>". Note this string must be used in a context where HTML escaping is done to avoid XSS vulnerabilities.

```
apex.lang.formatMessageNoEscape("CONFIRM", inputValue);
```

31.6.6 apex.lang.formatNoEscape

Same as format, except the replacement arguments are not HTML escaped. They must be known to be safe or are used in a context that is safe.

Parameters

Table 31-25 Parameters for apex.lang.formatNoEscape

Name	Type	Description
`pPattern`	String	The message pattern that contains one or more parameters %0 to %9.
`...*`	any	Optional replacement values one for each message parameter %0 to %9.

Returns

The localized and formatted message text string. If the key is not found then the key is returned.

Example

This example returns "You entered <ok>" when the inputValue variable value is "<ok>". Note this string must be used in a context where HTML escaping is done to avoid XSS vulnerabilities.

```
apex.lang.formatNoEscape("You entered %0", inputValue);
```

See Also:

"apex.lang.format (page 31-28)"

31.6.7 apex.lang.getMessage

Return the message associated with the given key. The key is looked up in the messages added with addMessages.

Parameters

Table 31-26 Parameters for apex.lang.getMessage

Name	Type	Description
pKey	String	The message key.

Returns

The localized message text string. If the key is not found then the key is returned.

Example

This example returns `"OK"` when the localized text for key `OK_BTN_LABEL` is `"OK"`.

```
apex.lang.getMessage("OK_BTN_LABEL");
```

31.7 apex.message namespace

The `apex.message` namespace is used to handle client-side display and management of messages in Oracle Application Express.

31.7.1 apex.message.addVisibilityCheck

This function is for APEX region plug-in developers. In order to navigate to items (page items or column items) that have an error (or anything else that can be in an error state), the error item must be visible before it is focused. Any region type that can possibly hide its contents should add a visibility check function using this method. Each function added is called for any region or item that needs to be made visible.

Syntax

```
apex.message.addVisibilityCheck( pFunction )
```

Parameters

Table 31-27 apex.message.addVisibilityCheck

Name	Type	Description
pFunction	Function	A function that is called with an element ID. The function should ensure that the element is visible if the element is managed or controlled by the region type that added the function.

Example

The following example explains a Region plug-in type called `Expander`, that can show or hide its contents and can contain page items. For purposes of example, this plug-in adds a `t-Expander` class to its region element and also has an `expand` method available, to expand its contents.

This region should register a visibility check function as follows:

```
apex.message.addVisibilityCheck( function( id ) {
   var lExpander$  = $( "#" + id ).closest( "t-Expander" );

   // Check if lExpander$ is an 'expander' region
   if ( lExpander$.hasClass( "t-Expander" ) ) {

       // If so, expander region must show its contents
       lExpander$.expander( "expand" );
   }
});
```

31.7.2 apex.message.alert

This function displays an alert dialog with the given message and OK button. The callback function passed as the `pCallback` parameter is called when the dialog is closed. The dialog displays using the jQuery UI `Dialog` widget.

There are some differences between this function and a browser's built-in alert function:

- The dialog style will be consistent with the rest of the application.
- The dialog can be moved.
- The call to `apex.message.alert` does not block. Any code defined following the call to `apex.message.alert` will run before the user presses **OK**. Therefore code to run after the user closes the dialog must be done from within the callback, as shown in the example.

Note:

If either of the following two pre requisites are not met, the function falls back to using the browser's built-in alert:

- jQuery UI dialog widget code must be loaded on the page.
- The browser must be running in **Standards** mode. This is because if it is running in **Quirks** mode (as is the case with some older themes), this can cause issues with display position, where the dialog positions itself in the vertical center of the page, rather than the center of the visible viewport.

Syntax

```
apex.message.alert( pMessage, pCallback )
```

Parameters

Table 31-28 apex.message.alert

Name	Type	Description
`pMessage`	String	The message to display in the confirmation dialog.
`pCallback`	Function	Callback function called when dialog is closed.

Example

The following example displays an alert **Load complete**, then after the dialog closes executes the `afterLoad()` function.

This region should register a visibility check function as follows:

```
 apex.message.alert( "Load complete.", function(){
    afterLoad();
});
```

31.7.3 apex.message.clearErrors

This function clears all the errors currently displayed on the page.

Syntax

```
apex.message.clearErrors()
```

Parameters

None

Example

The following example demonstrates clearing all the errors currently displayed on the page.

```
 apex.message.clearErrors();
```

31.7.4 apex.message.confirm

This function displays a confirmation dialog with the given message and OK and Cancel buttons. The callback function passed as the `pCallback` parameter is called when the dialog is closed, and passes `true` if OK is pressed and false otherwise. The dialog displays using the jQuery UI `Dialog` widget.

There are some differences between this function and a browser's built-in alert function:

- The dialog style will be consistent with the rest of the application.
- The dialog can be moved.
- The call to `apex.message.confirm` does not block. Any code defined following the call to `apex.message.confirm` will run before the user presses **OK** or **Cancel**. Therefore acting on the user's choice must be done from within the callback, as shown in the example.

Note:

If either of the following two pre requisites are not met, the function falls back to using the browser's built-in confirm:

- jQuery UI dialog widget code must be loaded on the page.
- The browser must be running in **Standards** mode. This is because if it is running in **Quirks** mode (as is the case with some older themes), this can cause issues with display position, where the dialog positions itself in the vertical center of the page, rather than the center of the visible viewport.

Syntax

```
apex.message.confirm( pMessage, pCallback )
```

Parameters

Table 31-29 apex.message.confirm

Name	Type	Description
`pMessage`	String	The message to display in the confirmation dialog.
`pCallback`	Function	Callback function called when dialog is closed. Function passes the following parameter: • OK Pressed: `true` if **OK** is pressed, `false` otherwise (if **Cancel** pressed, or the dialog was closed by some other means).

Example

The following example displays a confirmation message **Are you sure?**, and if **OK** is pressed executes the `deleteIt()` function.

This region should register a visibility check function as follows:

```
apex.message.confirm( "Are you sure?", function( okPressed ) {
    if( okPressed ) {
        deleteIt();
    }
});
```

31.7.5 apex.message.hidePageSuccess

This function hides the page-level success message.

> **Tip:**
>
> As a theme developer, you can influence or override what happens when hiding a page-level success message. For more information, please refer to the `apex.message.setThemeHooks` function (specifically the `beforeHide` callback function, where you would need to check for `pMsgType === apex.message.TYPE.SUCCESS` to isolate when hiding a page-level success message).

Syntax

```
apex.message.hidePageSuccess()
```

Parameters

None

Example

The following example demonstrates hiding the page-level success message.

```
apex.message.hidePageSuccess();
```

> **See Also:**
>
> apex.message.setThemeHooks (page 31-35)

31.7.6 apex.message.setThemeHooks

This function allows a theme to influence some behavior offered by the `apex.message` API. You can call this function from theme page initialization code.

Syntax

```
apex.message.setThemeHooks( pOptions )
```

Parameters

Table 31-30 apex.message.setThemeHooks

Name	Type	Description
`pOptions`	Object	An object with `pOptions` properties described in the following table.

Table 31-31 pOptions properties

Name	Type	Description
`beforeShow`	Function	Callback function that will be called prior to the default show page-notification functionality. Optionally return `false` from the callback to completely override default show functionality. Callback passes the following parameters: • `pMsgType`: Identifies the message type. Use `apex.message.TYPE` to identify whether showing an error or success message • `pElement$`: jQuery object containing the element being shown.
`beforeHide`	Function	Callback function that will be called prior to the default hide page-notification functionality. Optionally return `false` from the callback to completely override default hide functionality. Callback passes the following parameters: • `pMsgType`: Identifies the message type. Use `apex.message.TYPE` to identify showing an error or success message • `pElement$`: jQuery object containing the element being hidden.

Table 31-31 (Cont.) pOptions properties

Name	Type	Description
`closeNotificationSelector`	String	jQuery selector to identify the close buttons in notifications, defaults to that used by Universal Theme (`button.t-Button–closeAlert`). May be required by custom themes if you still want to have APEX handle the hide logic, and where messaging contains a close notification button with a different class.

Example

The following example shows `beforeShow` and `beforeHide` callbacks defined. To add and remove an additional class `animate-msg` on the notification element, before the default show and hide logic. This will happen for success messages only by the quality of the check on `pMsgType`.

> **Note:**
>
> The callbacks do not return anything, therefore the default show / hide behavior happens after the callback.

```
apex.message.setThemeHooks(
    beforeShow: function( pMsgType, pElement$ ){
        if ( pMsgType === apex.message.TYPE.SUCCESS ) {
            pElement$.addClass( "animate-msg" );
        }
    },
    beforeHide: function( pMsgType, pElement$ ){
        if ( pMsgType === apex.message.TYPE.SUCCESS ) {
            pElement$.removeClass( "animate-msg" );
        }
    }
});
```

31.7.7 apex.message.showErrors

This function displays all errors on the `apex.message` error stack. If you do not want to add to the stack, you must first call `clearErrors()`. Errors display using the current app's theme's templates. For page level messages (where location = `page`), error messages use markup from the page template's **Subtemplate > Notification** attribute. For item level messages (where location = `inline`), error messages use markup from the item's label template's **Error Display > Error Template** attribute.

Tip:

Theme developers must ensure the following:

- To display errors for your theme, it must define both of the template attributes described above. In addition, for `inline` errors the label template must reference the `#ERROR_TEMPLATE#` substitution string in either the **Before Item** or **After Item** attributes of your label templates.
- As a theme developer, you can influence or override what happens when showing page-level errors. For more information, please refer to the `apex.message.setThemeHooks` function (specifically the `beforeShow` callback function, where you would need to check for '`pMsgType === apex.message.TYPE.ERROR`' to isolate when showing page level errors).

Syntax

```
apex.message.showErrors( pErrors )
```

Parameters

Table 31-32 apex.message.showErrors

Name	Type	Description
`pErrors`	Object or Array	An object, or array of objects with properties described in the following table

Table 31-33 pErrors

Name	Type	Description
`type`	String	Must pass `error`, although may support different notification types in the future.
`location`	String | Array	Possible values are: `inline`, `page` or `[inline,page]`.
`pageItem`	String	Item reference where an `inline` error should display. Required when location = `inline`.
`message`	String	The error message.
`unsafe`	Boolean	Pass `true` so that the message will be escaped by `showErrors`. Pass `false` if the message is already escaped and does not need to be escaped by `showErrors`.

Examples

The following example first clears any existing errors displayed, and then displays two errors. To display the two errors, an array containing two error objects is passed to the `showErrors` call. The first error message is `Name is required!` and displays at both `page` level and `inline` with the item `P1_ENAME`. The second error messages is `Page error has occurred!` and displays just at `page` level. In both errors, the message text is considered safe and therefore will not be escaped.

```
// First clear the errors
apex.message.clearErrors();

// Now show new errors
apex.message.showErrors([
    {
        type:       "error",
        location:   [ "page", "inline" ],
        pageItem:   "P1_ENAME",
        message:    "Name is required!",
        unsafe:     false
    },
    {
        type:       "error",
        location:   "page",
        message:    "Page error has occurred!",
        unsafe:     false
    }
]);
```

See Also:

- "apex.message.setThemeHooks (page 31-35)"
- "apex.message.clearErrors (page 31-33)"

31.7.8 apex.message.showPageSuccess

This function displays a page-level success message. This clears any previous success messages displayed, and also assumes there are no errors and clears any errors previously displayed. Success messages display using the current app's theme's template. Specifically for page success messages, the markup from the page template's **Subtemplate > Success Message** attribute is used.

Tip:

As a theme developer, you can influence or override what happens when showing a page-level success message. For more information, refer to the `apex.message.setThemeHooks` function (specifically the `beforeShow` callback function, where you would need to check for `pMsgType === apex.message.TYPE.SUCCESS` to isolate when showing a page-level success message).

Syntax

```
apex.message.showPageSuccess( pMessage )
```

Parameters

Table 31-34 apex.message.showPageSuccess

Name	Type	Description
pMessage	String	The success message to display.

Example

The following example demonstrates a page-level success message `Changes saved!`.

```
apex.message.showPageSuccess( "Changes saved!" );
```

See Also:

apex.message.setThemeHooks (page 31-35)

31.7.9 apex.message.TYPE

The `TYPE` property is an object that contains properties (constants) that can be useful when calling some `apex.message` APIs. For example `setThemeHooks`.

Properties

Table 31-35 apex.message.TYPE properties

Value	Description
`SUCCESS: "success"`	Identifies a success message
`ERROR: "error"`	Identifies an error message

Example

See apex.message.setThemeHooks (page 31-35) for an example of how this can be used.

31.8 apex.navigation namespace

The `apex.navigation` namespace contains popup and redirect related functions of Oracle Application Express.

31.8.1 apex.navigation.dialog

Opens the specified page in a dialog. For mobile UI, the page is loaded using a role 'dialog' in a `mobile.changePage` call. For desktop UI, a modal page is loaded in an iframe using jQuery UI dialog widget. For desktop UI, a non-modal page is loaded in a popup browser window. The names `_self`, `_parent`, and `_top` should not be used. The window name is made unique so that it cannot be shared with other apps. Every effort is made to then focus the window.

Note:

Typically this API call is generated by the server when the page target is a modal page or by using `APEX_UTIL.PREPARE_URL`. At a minimum the url of the dialog page must be generated on the server so that the correct dialog checksum can be generated.

Parameters

Table 31-36 apex.navigation.dialog

Name	Type	Description
`pUrl`	String	The url of the page to load.

Table 31-36 (Cont.) apex.navigation.dialog

Name	Type	Description
`pOptions`	Object	Object to identify the attributes of the dialog, with the following properties: • title - the title of the dialog. The default is the name of the page. • height - height of dialog content area, in pixels, default is 500. • width - width of window content area, in pixels, default 500. • maxWidth - maximum width of window content area, in pixels, default 1500. • modal - true or false. Default is true. • dialog_attributes - optional attribute, to allow the setting of any additional options supported by the underlying dialog implementation. For example, to define jQuery UI Dialog attribute resizable: `resizable:false` **See Also:** See jQuery UI documentation of Dialog widget for all other available options for a modal dialog in a desktop user interface. `http://api.jqueryui.com/` See jQuery Mobile documentation of Dialog widget for all other available options for a modal dialog in a mobile user interface. `http://jquerymobile.com/`

Table 31-36 (Cont.) apex.navigation.dialog

Name	Type	Description
pCssClasses	String	To identify the CSS classes, if any, to be applied to the dialog, and appended on to the dialogClass attribute.
pTriggeringElement	String	jQuery selector to identify APEX page element opening the dialog.

Example

```
apex.navigation.dialog(url, {
    title:'About',
    height:'480',
    width:'800',
    maxWidth:'1200',
    modal:true,
    resizable:false },
    'a-Dialog--uiDialog',
    '#myregion_static_id');
```

31.8.2 apex.navigation.dialog.cancel

Closes the dialog window.

Parameters

Table 31-37 apex.navigation.dialog.cancel

Name	Type	Description
pIsModal	Boolean	To identify whether the dialog is modal.

31.8.3 apex.navigation.dialog.close

Executes an action and then closes the dialog window.

Parameters

Table 31-38 apex.navigation.dialog.close

Name	Type	Description
pIsModal	Boolean	To identify whether the dialog is modal.

Table 31-38 (Cont.) apex.navigation.dialog.close

Name	Type	Description
pAction	String, Function, Object	This can be: • a URL which will trigger a redirect in the parent page • a function to redirect to a different dialog page • false to cancel the dialog • an object of page items and values which will be exposed in the 'Dialog Closed dynamic action event

Example

To handle chaining from one modal dialog page to another:

```
apex.navigation.dialog.close(true, function( pDialog ) {
    apex.navigation.dialog(url, {
        title:'About',
        height:'480',
        width:'800',
        maxWidth:'1200',
        modal:true,
        dialog:pDialog,
        resizable:false },
        'a-Dialog--uiDialog',
        '#myregion_static_id');

});
```

31.8.4 apex.navigation.dialog.fireCloseHandler

Fires the internal `close` event of a dialog which was registered with the `registerCloseHandler` when the dialog was opened.

Parameters

Table 31-39 apex.navigation.dialog.fireCloseHandler

Name	Type	Description
`pOptions`	Object	pOptions has to contain the following attributes: • `"handler$"` jQuery object where the event will be registered for. • `"dialog"` DOM/ jQuery/... object of the current dialog instance which will be passed into the open dialog call if the existing dialog should be re-used. • `"closeFunction"` Function which is used to close the dialog.

31.8.5 apex.navigation.dialog.registerCloseHandler

Registers the internal `"close"` event of a dialog. The event will be triggered by `fireCloseEvent` and depending on the passed in `pAction`, it will:

- Re-use the existing dialog and navigate to a different dialog page
- Navigate to a different page in the caller
- Cancel the dialog
- Close the dialog and trigger the `"apexafterclosedialog"` event

Parameters

Table 31-40 apex.navigation.dialog.registerCloseHandler

Name	Type	Description
`pOptions`	Object	`pOptions` has to contain the following attributes : • `"handler$"` jQuery object where the event will be registered for. • `"dialog"` DOM/ jQuery/... object of the current dialog instance which will be passed into the open dialog call if the existing dialog should be re-used. • `"closeFunction"` Function which is used to close the dialog.

31.8.6 apex.navigation.openInNewWindow

Opens the given url in a new named window or tab (the browser / browser user preference settings may control if a window or tab is used). If a window with that name already exists it is reused. The names _self, _parent and _top should not be used. The window name is made unique so that it cannot be shared with other apps. Every effort is made to then focus the window.

Note:

Firefox and IE will not focus a tab if that tab is not the currently active tab in its browser window. This is why, unless `favorTabbedBrowsing` is `true`, this API forces the url to open in a new window so that it has a better chance of being focused.

Note:

For Opera, the Advanced/content > JavaScript Options: “Allow raising of windows" must be checked in order for focus to work.

Note:

To avoid being suppressed by a popup blocker, call this from a click event handler on a link or button.

Parameters

Table 31-41 apex.navigation.openInNewWindow

Name	Type	Description
`pUrl`	String	The url of the page to load.
`pWindowName`	String	The name of the window (optional) The default is `"_blank"`
`pOptions`	Object	Optional object with the following properties: • `favorTabbedBrowsing` - {Boolean} if true don't try to force a new window for the benefit of being able to focus it. This option only affects Firefox and IE.

Returns

Returns the window object of the named window or null if for some reason the window isn't opened.

Example

```
apex.navigation.openInNewWindow(url, "MyWindow");
```

31.8.7 apex.navigation.popup

Opens the given url in a new typically named popup window. If a window with that name already exists it is reused. If no name is given or the name is "_blank" then a new unnamed popup window is opened. The names _self, _parent and _top should not be used. The window name is made unique so that it cannot be shared with other applications.

Note:

To avoid being suppressed by a popup blocker, call this from a click event handler on a link or button.

Parameters

Table 31-42 apex.navigation.popup

Name	Type	Description
pOptions	Object	An object with the following optional properties: • `url` - the page url to open in the window. The default is `"about:blank"`. • `name` - the name of the window. The default is `"_blank"`, which opens a new unnamed window. • `height` - height of window content area in pixels. Default 600. • `width` - width of window content area in pixels. Default 600. • `scroll` - `"yes"` or `"no"` Default is `"yes"`. • `resizeable` - `"yes"` or `"no"` . Default is `"yes"`. • `toolbar` - `"yes"` or `"no"`. Default is `"no"`. • `location` - `"yes"` or `"no"`. Default is `"no"`. • `statusbar` - `"yes"` or `"no"`. Default is `"no"`. This controls the status feature. • `menubar` - `"yes"` or `"no"`. Default is `"no"`.

Example

```
apex.navigation.popup ({
    url: "about:blank",
     name: "_blank",
     width: 400,
     height: 400,
     scroll: "no",
     resizable: "no",
     toolbar: "yes" });
```

31.8.8 apex.navigation.popup.close

Sets the value of the item (`pItem`) in the parent window, with (`pValue`) and then closes the popup window.

Parameters

Table 31-43 apex.navigation.popup.close

Name	Type	Description
`pItem`	DOM node | string ID	The item to set.
`pValue`	string	The item value to set.

31.8.9 apex.navigation.redirect

Opens the specified page (`pWhere`) in the current window.

Parameters

Table 31-44 apex.navigation.redirect

Name	Type	Description
`pWhere`	String	The url of the page to open in the current window.

31.9 apex.page namespace

This namespace is used for all client-side page related functions of Oracle Application Express.

31.9.1 apex.page.cancelWarnOnUnsavedChanges

Call to remove the handler that checks for unsaved changes. This is useful to do before any kind of cancel operation where the user is intentionally choosing to loose the changes. It is not normally necessary to call this function because the declarative attribute `Warn on Unsaved Changes` with value `Do Not Check` will do it automatically. Adding the class *js-ignoreChange* to a link (anchor element) or button will cause this function to be called before the link or button action.

Parameters

None.

Example

The following sets up a handler on a custom cancel button that leaves the page without checking for changes.

```
apex.jQuery( "#custom-cancel-button" ).click( function() {
    apex.page.cancelWarnOnUnsavedChanges();
    apex.navigation.redirect( someUrl );
});
```

31.9.2 apex.page.confirm Signature 1

Displays a confirmation dialog showing a message, `pMessage`, and depending on user's choice, submits the page setting the request value to `pRequest`, or does not submit the page. Once the user chooses to submit the page the behavior is the same as for the `apex.page.submit` function. The shorter alias for this function `apex.confirm` with the same parameters can also be used.

Parameters

Table 31-45 apex.page.confirm Signature 1 Parameters

Name	Type	Description
`pMessage`	String	The confirmation message to display in the dialog.
`pRequest`	String	The request value.

Example

This example shows a confirmation dialog with the text 'Delete Department'. If the user chooses to proceed with the delete, the current page is submitted with a REQUEST value of 'DELETE'.

```
apex.page.confirm('Delete Department', 'DELETE');
```

or

```
apex.confirm('Delete Department', 'DELETE');
```

31.9.3 apex.page.confirm Signature 2

Displays a confirmation dialog showing a message,`pMessage`, and depending on user's choice, submits the page according to the options in `pRequest`, or does not submit the page. Once the user chooses to submit the page the behavior is the same as for the `apex.page.submit` function. The shorter alias for this function `apex.confirm` with the same parameters can also be used.

Parameters

Table 31-46 apex.page.confirm Signature 2 Parameters

Name	Type	Description
`pMessage`	String	The confirmation message to display in the dialog.

Table 31-46 (Cont.) apex.page.confirm Signature 2 Parameters

Name	Type	Description
pOptions	Object	Options to control how the page is submitted. See "poptions properties" in apex.page.submit Signature 2 (page 31-52).

Example

This example shows a confirmation message with the 'Save Department?' text. If the user chooses to proceed with the save, the page is submitted with a REQUEST value of 'SAVE' and 2 page item values are set, P1_DEPTNO to 10 and P1_EMPNO to 5433.

```
apex.page.confirm("Save Department?", {
  request:"SAVE",
  set:{"P1_DEPTNO":10, "P1_EMPNO":5433}
  });
```

or

```
apex.confirm("Save Department?", {
  request:"SAVE",
  set:{"P1_DEPTNO":10, "P1_EMPNO":5433}
  });
```

31.9.4 apex.page.submit Signature 1

This function submits the page with the Application Express REQUEST value set to pRequest. The shorter alias for this function apex.submit with the same parameters can also be used. Depending on the value of the page's Reload on Submit attribute the page is submitted using ajax or with a normal form submission post request.

This function triggers a "apexbeforepagesubmit" event on the apex.gPageContext$ which can be canceled. If canceled the page is not submitted.

Just before the page is submitted this function triggers a "apexpagesubmit" event on the apex.gPageContext$ which cannot be canceled.

Parameters

Table 31-47 apex.page.submit Signature 1 Parameters

Name	Type	Description
pRequest	String	The request value.

Example

Submits the current page with a REQUEST value of 'DELETE'.

```
apex.page.submit( 'DELETE' );
```

or

```
apex.submit( 'DELETE' );
```

31.9.5 apex.page.submit Signature 2

This function submits the page using the options specified in `pOptions`. The shorter alias for this function `apex.submit` with the same parameters can also be used. Depending on the value of the page's Reload on Submit attribute the page is submitted using ajax or with a normal form submission post request.

This function triggers a "`apexbeforepagesubmit`" event on the `apex.gPageContext$` which can be canceled. If canceled the page is not submitted.

Just before the page is submitted this function triggers a "`apexpagesubmit`" event on the `apex.gPageContext$` which cannot be canceled.

Parameters

Table 31-48 apex.page.submit Signature 2 Parameters

Name	Type	Description
`pOptions`	Object	Options to control how the page is submitted. See Table pOptions for each option property.

Table 31-49 pOptions properties

Name	Description
`request`	The `REQUEST` value. For a confirm function the default is "`Delete`" for a submit function the default is `null`.
`set`	An object containing name/value pairs of items to set on the page prior to submission. The object properties are page item names and the item value is set to the property value. The default is to not set any page items.
`showWait`	If `true` a '`Wait Indicator`' spinner is displayed, which can be useful when running long page operations. The defaults is `false`.
`submitIfEnter`	If you only want to submit when the `ENTER` key has been pressed, call `apex.submit` in the event callback and pass the event object as this parameter.
`reloadOnSubmit`	Override the Reload on Submit setting of the page. One of "`A`" (always) or "`S`" (success).

Table 31-49 (Cont.) pOptions properties

Name	Description
ignoreChange	If `true` (the default) and the `warnOnUnsavedChanges` feature is enabled no warning will be given if there are changes. If `false` and the `warnOnUnsavedChanges` feature is enabled and there are changes there will be a warning. If `warnOnUnsavedChanges` feature is disabled there is never a warning.
validate	If `true` check the validity of page items and models before submitting the page. If anything is not valid then show an alert dialog and don't submit the page. The default is `false`.

Returns

If the `submitIfEnter` option is specified, a Boolean value is returned. If the ENTER key is not pressed, `true` is returned and if the `ENTER` key was pressed `false` is returned. If `submitIfEnter` is not been specified, nothing is returned.

Example

This example submits the page with a `REQUEST` value of '`DELETE`' and 2 page item values are set, `P1_DEPTNO` to `10` and `P1_EMPNO` to `5433` . During submit a wait icon is displayed as visual indicator for the user as well.

```
apex.page.submit({
    request:"DELETE",
    set:{"P1_DEPTNO":10, "P1_EMPNO":5433},
    showWait: true
});
```

or

```
apex.submit({
    request:"DELETE",
    set:{"P1_DEPTNO":10, "P1_EMPNO":5433},
    showWait: true
});
```

31.9.6 apex.page.validate

Check if page items and submittable Application Express models on the page are valid. Any errors are shown inline using `apex.message.showErrors` function.

Parameters

None.

Returns

`true` if the page is valid and `false` otherwise.

Example

The following example checks if the page is valid when a button with id `checkButton` is pressed.

```
apex.jQuery( "#checkButton" ).click( function() {
    if ( !apex.page.validate() ) {
        alert("Please correct errors");
    }
});
```

31.9.7 apex.page.isChanged

Return true if any page items or Application Express models on this page have changed since last being sent to the server. This will call the `pExtraIsChanged` function set in `apex.page.warnOnUnsavedChanges` if one was supplied and only if no other changes are found first.

Parameters

None.

Returns

`true` if any page items or Application Express models on the page have changed and `false` otherwise.

Example

The following example checks if the page is changed before performing some action.

```
If ( apex.page.isChanged() ) {
   // do something when the page has changed
}
```

31.9.8 apex.page.warnOnUnsavedChanges

Initialize a handler that checks for unsaved changes anytime the page is about to unload. This is safe to call multiple times. The `pMessage` and `pExtraIsChanged` override any previous values. This function is called automatically when the page attribute Warn on Unsaved Changes is set to yes. The main reason to call this manually is to customize the parameters.

Parameters

Table 31-50 apex.page.warnOnUnsavedChanges

Name	Type	Description
`pMessage`	String	Optional message to display in the warning dialog when there are unsaved changes. If not given a default message is used. **Note:** Not all browsers will show this message.
`pExtraIsChanged`	Function	Optional additional function to be called that checks if there are any unsaved changes. It should return `true` if there are unsaved changes and `false` otherwise. It is only called if there are no changes to any models or page items. This is useful if there are non-standard state-full inputs on the page that are not Application Express items and do not keep their state in an Application Express model. It allows writing a custom function to detect if those non-standard inputs have changed.

Example

The following example enables the warn on unsaved changes feature with a custom message.

```
apex.page.warnOnUnsavedChanges( "The employee record has been changed" );
```

31.10 apex.region

The `apex.region` API provides a single interface for all common region related functionality of Application Express. This API returns an Application Express region object, which is used to access region related functions and properties.

apex.region (page 31-56)

apex.region.create (page 31-58)

apex.region.destroy (page 31-59)

apex.region.isRegion (page 31-59)

apex.region.findClosest (page 31-60)

31.10.1 apex.region

This API returns an Application Express region object for the given region id. The returned region object can then be used to access region related functions and properties.

Plug-in developers can define the behavior of their region by calling `apex.region.create`.

Parameters

Table 31-51 apex.region

Name	Type	Description
`pRegionId`	String	Region id or region static id. It is a best practice to give a region a Static ID if it is going to be used from JavaScript otherwise an internally generated id is used. The region id is substituted in the region template using the `#REGION_STATIC_ID#` string. The region id can be found by viewing the page source in the browser.

Returns

A region object is returned or null if there is no DOM element with an id equal to pRegionId. The region object has these properties and methods:

- focus (page 31-57)
- refresh (page 31-57)
- widget (page 31-57)

It may have additional properties or functions depending on the specific region. It is up to the region plug-in to document any additional properties or functions.

Example

This will not be used by itself, rather it is used to access item specific functions and properties, as documented in the following APIs.

See Also:

"apex.region.create (page 31-58)"

focus (page 31-57)

31.10.1.1 focus

This region object function causes the Application Express region to take focus. Not all native or plug-in regions support taking focus. It is up to the specific region to determine where focus will go. The default behavior is to set focus to the first element in the region that is capable of being tabbed to. Some regions manage focus such that there is a single tab stop (or limited number of tab stops) and they may put focus to where the user last had focus within the region.

Parameters

None.

Example

The following example will focus the region with static id `"myRegion"`.

```
var region = apex.region( "myRegion" );
region.focus();
```

31.10.1.2 refresh

This region object function refreshes the Application Express region. Typically this involves getting updated content or data from the server. Not all native or plug-in regions support refresh.

Note:

This function should be used in place of the legacy way of refreshing a region which was to trigger the `apexrefresh` event on the region element. For regions that still work this old way the default implementation of refresh will trigger the `apexrefresh` event.

Parameters

None.

Example

The following example will refresh the region with static id `"myRegion"`.

```
var region = apex.region( "myRegion" );
region.refresh();
```

31.10.1.3 widget

This region object function returns the widget associated with the Application Express region or null if the region isn't implemented with a widget. Some advanced region types such as Calendar, Interactive Grid, or Tree are implemented using a widget. This function provides access to the widget typically by returning a jQuery object for the widget element. You can then call widget methods on the jQuery object.

Parameters

None.

Example

The following example adds a row to an interactive grid by using the region widget function to access an `interactiveGrid` widget and then invoking the add-row action.

```
apex.region( "myGridRegion" ).widget().interactiveGrid( "getActions" ).invoke( "add-
row" )
```

31.10.2 apex.region.create

This function is only for region plug-in developers. It provides a plug-in specific implementation for the region.

Parameters

Table 31-52 apex.region.create

Name	Type	Description
`pRegionId`	String	Region id or region static id. It is a best practice to give a region a Static ID if it is going to be used from JavaScript otherwise an internally generated id is used. The region id is substituted in the region template using the `#REGION_STATIC_ID#` string. The region id can be found by viewing the page source in the browser.
`pRegionImpl`	Object	An object that provides the functions and properties for the region. It should provide a string type property. It can provide any additional functions that would be useful to developers. A default implementation is provided for any standard functions or properties omitted.

Example

The following is an example of what the region initialization code for a hypothetical region plug-in would do.

```
function initFancyList( pRegionId, … ) {
    …
    apex.region.create( pRegionId, {
```

```
        type: "FancyList",
        focus: function() {
            /* code to focus region */
        },
        refresh: function() {
            /* code to refresh region */
        }
    });
}
```

31.10.3 apex.region.destroy

This function is only for region plug-in developers. It will destroy and remove the behaviors associated with a region element. It does not remove the region element from the DOM. It is not necessary to call this function if the region will exist for the lifetime of the page. If the region is implemented by a widget that has a destroy method then this function can be called when the widget is destroyed.

Parameters

Table 31-53 apex.region.destroy

Name	Type	Description
`pRegionId`	String	Region id or region static id. It is a best practice to give a region a Static ID if it is going to be used from JavaScript otherwise an internally generated id is used. The region id is substituted in the region template using the `#REGION_STATIC_ID#` string. The region id can be found by viewing the page source in the browser.

Example

The following is an example of when the region is destroyed but the region element will remain on the page call:

```
apex.region.destroy( someRegionId );
```

31.10.4 apex.region.isRegion

This function returns true if and only if there is a DOM element with id equal to `pRegionId` that has had a region created for it with `apex.region.create`. To support older regions that don't implement a region interface (by calling `apex.region.create`) the default implementation of `apex.region` will attempt to treat any DOM element with an id as if it were an Application Express region. This function allows you to distinguish true Application Express regions from arbitrary DOM elements.

Parameters

Table 31-54 apex.region.isRegion

Name	Type	Description
`pRegionId`	String	Region id or region static id. It is a best practice to give a region a Static ID if it is going to be used from JavaScript otherwise an internally generated id is used. The region id is substituted in the region template using the `#REGION_STATIC_ID#` string. The region id can be found by viewing the page source in the browser.

Returns

`true` if `pRegionId` is the id of an Application Express region that implements the region interface and `false` otherwise.

Example

The following example will only focus the region if it is an Application Express region.

```
if ( apex.region.isRegion( someId ) ) {
    apex.region( someId ).focus();
}
```

31.10.5 apex.region.findClosest

Returns the region that contains the `pTarget` element. Returns null if there is no `pTarget` element or if it is not in a region that has been initialized with a call to `apex.region.create`.

Parameters

Table 31-55 apex.region.findClosest

Name	Type	Description
`pTarget`	DOM Element | String	A DOM element or CSS selector suitable as the first argument to the jQuery function.

Returns

A region object is returned or null if the element corresponding to `pTarget` is not inside a region. See the return value of `apex.region` for details about the region object.

Example

The following example will refresh the region that contains a button with class "`.refresh-button`" when it is clicked.

```
apex.jQuery( ".refresh-button" ).click( function( event ) {
    var region = apex.region.findClosest( event.target );
    if ( region ) {
        region.refresh();
    }
});
```

31.11 apex.server namespace

The `apex.server` namespace stores all Ajax functions to communicate with the server part of Oracle Application Express.

31.11.1 apex.server.loadScript

This API is used to asynchronously load other JavaScript libraries that has Asynchronous Module Definition (AMD), and if `RequireJS` library is already loaded on the page.

Syntax

```
apex.server.loadScript( pConfig, callback )
```

Parameters

Table 31-56 apex.server.loadScript

Name	Type	Description
`pConfig`	object	JSON objects contains the following attributes: • `"path"` - {String} the location of the file. • `"requirejs"` - {Boolean} whether to use `requirejs` to load this file. • `"global"` - {String} the global name introduced by this files and the existing one is overwritten. Leave this option empty if the file is generated by `RequireJS` optimizer.
`callback`	object	Function to be executed once script is loaded.

Returns

Returns a function.

Examples

The following example loads a regular library that does not need `RequireJS`:

```
apex.server.loadScript ({
   "path": "./library_1.js"
}, function() {
   console.log( 'library_1 is ready.' );
});
```

The following example loads a library that requires `RequireJS` and creates own namespace: `"myModule"`:

```
apex.server.loadScript ({
   "path": "./library_2.js",
   "requirejs": true,
   "global": "myModule"
}, function() {
   console.log( myModule );
});
```

The following example loads a concatenated libraries generated by `RequireJS` Optimizer, assuming `requireJS` is already on the page:

```
apex.server.loadScript ({
   "path": "./library_all.js",
   "requirejs": true
}, function() {
```

```
    console.log( myModule_1, myModule_2 ... );
});
```

31.11.2 apex.server.plugin

This function calls the PL/SQL Ajax function that has been defined for a plug-in. This function is a wrapper of the `jQuery.ajax` function that supports a subset of the `jQuery.ajax` options plus additional Application Express specific options.

The plug-in PL/SQL Ajax function is identified using the value returned by the PL/SQL package `apex_plugin.get_ajax_identifier`. There are two ways to provide the plug-in Ajax identifier:

1. Provide the `pAjaxIdentifier` as the first argument
2. Provide information about the region(s) including the `ajaxIdentifier` in the `pData` object structure. See `pData` description for details.

Syntax

```
apex.server.plugin( pAjaxIdentifier, pData, pOptions ) → Promise
```

Parameters

Table 31-57 apex.server.plugin

Name	Type	Description
`pAjaxIdentifier`	String	Optional. The plug-in ajax identifier. If not given then `pData` must include a regions array that includes a region with property `ajaxIdentifier`.
`pData`	Object	Optional object containing data to send to the server in the Ajax request. The Object is serialized as JSON and sent to the server in parameter `p_json`. See below `pData` special properties table for a list of properties that are treated special. Data for specific regions can be sent in the following format: `{ "regions": [ { "id": <region-id-or-static-id>, "ajaxIdentifier": <ajaxIdentifier>, <any other data specific to the region plug-in> }, ... ] }`

Table 31-57 (Cont.) apex.server.plugin

Name	Type	Description
`pOptions`	Object	Optional object that is used to set additional options to control the Ajax call including before and after processing. See below `pOptions` properties table for a list of Application Express specific options. See jQuery documentation of jQuery.ajax for these supported options: `accepts`, `dataType`, `beforeSend`, `contents`, `converters`, `dataFilter`, `headers`, `complete`, `statusCode`, `error`, `success`. The dataType option defaults to json. The async option is deprecated and will be removed in a future release. `http://docs.jquery.com/`

Table 31-58 pData special properties

Name	Type	Description
`pageItems`	String | jQuery | DOM element | Array	Identifies the page or column items that will be included in the request. It can be a jQuery selector, jQuery or DOM object or array of item names. These items will be made available in session state on the server. If `pageItems` contains column items then `pOptions` should include the target property so that the region session state context can be determined.
`x01 - x10`	String	These properties are moved out of the `p_json` object and sent as `x01 - x10` scalar parameters.
`f01 - f20`	String | Array	These properties are moved out of the `p_json` object and sent as `f01 - f20` array parameters.

Table 31-59 pOptions properties

Name	Type	Description
`refreshObject`	String | jQuery | DOM element	jQuery selector, jQuery or DOM object which identifies the DOM element on which the `apexbeforerefresh` and `apexafterrefresh` events are triggered. If this option is not supplied these events are not triggered.
`refreshObjectData`	Object | Array	Only applies if `refreshObject` option is given. Specify data that is passed by the `apexbeforerefresh` and `apexafterrefresh` event triggering code, so that any handlers defined for these events can access this data. In Dynamic Actions defined for the `Before Refresh` or `After Refresh` events, this can be accessed from JavaScript using the `this.data` property. For custom jQuery event handlers, this can be accessed through the `pData` parameter of the event handler.
`clear`	Function	JavaScript function used to clear the DOM after the `apexbeforerefresh` event is triggered and before the actual Ajax call is made.

Table 31-59 (Cont.) pOptions properties

Name	Type	Description
`loadingIndicator`	String | jQuery | DOM object | Function	Identifies the element(s) that will have a loading indicator (progress spinner) displayed next to it during the Ajax call. The element can be specified with a jQuery selector, jQuery or DOM object. `loadingIndicator` can also be a function which gets the loading Indicator as jQuery object and has to return the jQuery reference to the created loading indicator. For example: `function( pLoadingIndicator ) { return pLoadingIndicator.prependTo( apex.jQuery( "td.shuttleControl", gShuttle) ); }`
`loadingIndicatorPosition`	String	Six options to define the position of the loading indicator displayed. Only considered if the value passed to `loadingIndicator` is not a function. • `before`: Displays before the DOM element(s) defined by `loadingIndicator`. • `after`: Displays after the DOM element(s) defined by `loadingIndicator`. • `prepend`: Displays inside at the beginning of the DOM element(s) defined by `loadingIndicator`. • `append`: Displays inside at the end of the DOM element(s) defined by `loadingIndicator`. • `centered`: Displays in the center of the DOM element defined by `loadingIndicator`. • `page`: Displays in the center of the page.

Table 31-59 (Cont.) pOptions properties

Name	Type	Description
queue	Object	Object specifying the name of a queue and queue action. The name property specifies the name of the queue to add the request to. The action property can be one of the following: • "wait" This action is the default and is used to send requests one after the other. When the action is wait the request is added to the named queue. If there are no other requests in that queue in progress or waiting then this request is executed. Otherwise it waits on the named queue until the ones before it are complete. • "replace" Action replace is used when this current request makes any previous requests on the named queue in progress or waiting obsolete or invalid. This current request aborts any in progress request and clears out any waiting requests on the named queue and then is executed. Waiting requests are rejected with status "superseded". • "lazyWrite" This action is used to throttle requests to the server to persist data. This should only be used to persist non critical data such as user interface settings or state. Use when the data may change frequently and only the last data values need to be saved. For example this is useful for persisting splitter position, or tree expansion and focus state etc. The queue name is unique for each data unit. For example if you were saving the position of two different splitters use a unique name for each one so that latest update to one doesn't overwrite a previous lazy write of the other. When using lazyWrite Queue the refreshObject, clear, loadingIndicator, and loadingIndicatorPosition are most likely not useful because nothing is being loaded or refreshed. It is possible to mix requests with wait and replace actions on the same queue. The lazyWrite action should not be used with a queue name that is also used with wait and replace actions.
target	String | DOM element	Target element (DOM element or jQuery Selector) that this request pertains to. This is used to get session state context from the enclosing region.

Returns

Returns a promise object. The promise done method is called if the Ajax request completes successfully. This is called in the same cases as the success callback function in pOptions. The promise fail method is called if the Ajax request completes with an error including internally detected Application Express errors. This is called in the same cases as the error callback function in pOptions. The promise also has an always method that is called after done and error. The promise is returned even when queue options are used. The promise is not a jqXHR object but does have an abort method. The abort method does not work for requests that use any queue options.

Example

This example demonstrates a call to apex.server.plugin sets the scalar value x01 to test (which can be accessed from PL/SQL using apex_application.g_x01) and sets the page item's P1_DEPTNO and P1_EMPNO values in session state (using jQuery selector syntax). The P1_MY_LIST item is used as the element for which the

apexbeforerefresh and apexafterrefresh events are fired. P1_MY_LIST is used as the element for which to display the loading indicator next to. The success callback is stubbed out and is used for developers to add their own code that fires when the call successfully returns. The value for lAjaxIdentifier must be set to the value returned by the server PL/SQL API apex_plugin.get_ajax_identifier.

The pData parameter to the success callback will contain any response returned by the plug-in PL/SQL Ajax function.

```
apex.server.plugin ( lAjaxIdentifier, {
    x01: "test",
    pageItems: "#P1_DEPTNO,#P1_EMPNO"
    }, {
       refreshObject: "#P1_MY_LIST",
       loadingIndicator: "#P1_MY_LIST",
       success: function( pData ) {
         // do something here
       }
    });
```

31.11.3 apex.server.pluginUrl

Returns the URL to issue a GET request to the PL/SQL Ajax function which has been defined for a plug-in.

Parameters

Table 31-60 apex.server.pluginUrl

Name	Type	Description
pAjaxIdentifier	String	Use the value returned by the PL/SQL package apex_plugin.get_ajax_identifier to identify your plug-in.
pData	Object	Optional object that is used to set additional values which are included into the URL. The special attribute pageItems which can be of type jQuery selector, jQuery or DOM object or array of item names identifies the page items which are included in the URL. You can also set additional parameters that the apex.show procedure provides (for example you can set the scalar parameters x01 - x10 and the arrays f01 - f20).

Returns

Returns a string that is the URL to issue the GET request.

Example

This call to apex.server.pluginUrl returns a URL to issue a GET request to the PL/SQL Ajax function which has been defined for a plug-in, where the URL sets the scalar value x01 to test (which can be accessed from PL/SQL using apex_application.g_x01) and will also set the page item's P1_DEPTNO and P1_EMPNO values in session state (using jQuery selector syntax). The value for lAjaxIdentifier must be set to the value returned by the server PL/SQL API apex_plugin.get_ajax_identifier.

```
var lUrl = apex.server.pluginUrl ( lAjaxIdentifier, {
  x01: "test",
  pageItems: "#P1_DEPTNO,#P1_EMPNO" } );
```

31.11.4 apex.server.process

This function calls a PL/SQL on-demand (Ajax callback) process defined on page or application level. This function is a wrapper of the jQuery.ajax function that supports a subset of the `jQuery.ajax` options plus additional Application Express specific options.

Syntax

```
apex.server.process( pName, pData, pOptions ) → Promise
```

Parameters

Table 31-61 apex.server.process

Name	Type	Description
`pName`	String	The name of the PL/SQL on-demand page or application process to call.

`pData`	Object	Optional object containing data to send to the server in the Ajax request. The Object is serialized as JSON and sent to the server in parameter p_json. See below `pData` table for a list of properties that are treated special.
		Data for specific regions can be sent in the following format:
		`{`
		`"regions": [ {`
		`"id": <region-id-or-static-id>,`
		`"ajaxIdentifier": <ajaxIdentifier>,`
		`<any other data specific to the region plug-in>`
		`}, ...`
		`]`
		`}`

`pOptions`	Object	Optional object that is used to set additional options to control the Ajax call including before and after processing. See `pOptions` table below for a list of Application Express specific options. See jQuery documentation of `jQuery.ajax` for these supported options: `accepts`, `dataType`, `beforeSend`, `contents`, `converters`, `dataFilter`, `headers`, `complete`, `statusCode`, `error`, `success`. The dataType option defaults to json. The async option is deprecated and will be removed in a future release.

Table 31-62 pData special properties

Name	Type	Description
`pageItems`	String | jQuery | DOM element | Array	Identifies the page or column items that will be included in the request. It can be a jQuery selector, jQuery or DOM object or array of item names. These items will be made available in session state on the server. If `pageItems` contains column items then `pOptions` should include the target property so that the region session state context can be determined.
`x01 - x10`	String	These properties are moved out of the `p_json` object and sent as `x01 - x10` scalar parameters.
`f01 - f20`	String | Array	These properties are moved out of the `p_json` object and sent as `f01 - f20` array parameters.

Table 31-63 pOptions properties

Name	Type	Description
`refreshObject`	String | jQuery | DOM element	jQuery selector, jQuery or DOM object which identifies the DOM element on which the `apexbeforerefresh` and `apexafterrefresh` events are triggered. If this option is not supplied these events are not triggered.
`refreshObjectData`	Object | Array	Only applies if `refreshObject` option is given. Specify data that is passed by the `apexbeforerefresh` and `apexafterrefresh` event triggering code, so that any handlers defined for these events can access this data. In Dynamic Actions defined for the Before Refresh or After Refresh events, this can be accessed from JavaScript using the `this.data` property. For custom jQuery event handlers, this can be accessed through the `pData` parameter of the event handler.
`clear`	Function	JavaScript function used to clear the DOM after the `apexbeforerefresh` event is triggered and before the actual Ajax call is made.

Table 31-63 (Cont.) pOptions properties

Name	Type	Description
`loadingIndicator`	String | jQuery | DOM object | Function	Identifies the element(s) that will have a loading indicator (progress spinner) displayed next to it during the Ajax call. The element can be specified with a jQuery selector, jQuery or DOM object. `loadingIndicator` can also be a function which gets the loading Indicator as jQuery object and has to return the jQuery reference to the created loading indicator. For example: `function( pLoadingIndicator ) { return pLoadingIndicator.prependTo( apex.jQuery( "td.shuttleControl", gShuttle) ); }`
`loadingIndicatorPosition`	String	Six options to define the position of the loading indicator displayed. Only considered if the value passed to `loadingIndicator` is not a function. `loadingIndicatorPosition` - Six options to define the position of the loading indicator displayed. Only considered if the value passed to `loadingIndicator` is not a function. • `before`: Displays before the DOM element(s) defined by `loadingIndicator` • `after`: Displays after the DOM element(s) defined by `loadingIndicator` • `prepend`: Displays inside at the beginning of the DOM element(s) defined by `loadingIndicator` • `append`: Displays inside at the end of the DOM element(s) defined by `loadingIndicator` • `centered`: Displays in the center of the DOM element defined by `loadingIndicator` • `page`: Displays in the center of the page.

Table 31-63 (Cont.) pOptions properties

Name	Type	Description
queue	Object	Object specifying the name of a queue and queue action. The name property specifies the name of the queue to add the request to. The action property can be one of the following: • `"wait"` This action is the default and is used to send requests one after the other. When the action is wait the request is added to the named queue. If there are no other requests in that queue in progress or waiting then this request is executed. Otherwise it waits on the named queue until the ones before it are complete. • `"replace"` Action replace is used when this current request makes any previous requests on the named queue in progress or waiting obsolete or invalid. This current request aborts any in progress request and clears out any waiting requests on the named queue and then is executed. Waiting requests are rejected with status `"superseded"`. • `"lazyWrite"` This action is used to throttle requests to the server to persist data. This should only be used to persist non critical data such as user interface settings or state. Use when the data may change frequently and only the last data values need to be saved. For example this is useful for persisting splitter position, or tree expansion and focus state etc. The queue name is unique for each data unit. For example if you were saving the position of two different splitters use a unique name for each one so that latest update to one doesn't overwrite a previous lazy write of the other. When using `lazyWrite` Queue the `refreshObject`, clear, loadingIndicator, and `loadingIndicatorPosition` are most likely not useful because nothing is being loaded or refreshed. It is possible to mix requests with wait and replace actions on the same queue. The `lazyWrite` action should not be

Table 31-63 (Cont.) pOptions properties

Name	Type	Description
		used with a queue name that is also used with wait and replace actions.
`target`	String | DOM element	Target element (DOM element or jQuery Selector) that this request pertains to. This is used to get session state context from the enclosing region.

Returns

A promise object is returned. The promise done method is called if the Ajax request completes successfully. This is called in the same cases as the success callback function in `pOptions`. The promise fail method is called if the Ajax request completes with an error including internally detected Application Express errors. This is called in the same cases as the error callback function in `pOptions`. The promise also has an always method that is called after done and error. The promise is returned even when queue options are used. The promise is not a `jqXHR` object but does have an abort method. The abort method does not work for requests that use any queue options.

Example

This example demonstrates a call to `apex.server.process` calls an on-demand process called `MY_PROCESS` and sets the scalar value `x01` to `test` (which can be accessed from PL/SQL using `apex_application.g_x01`) and sets the page item's `P1_DEPTNO` and `P1_EMPNO` values in session state (using jQuery selector syntax). The success callback is stubbed out so that developers can add their own code that fires when the call successfully returns.

Note: The `pData` parameter to the success callback contains the response returned from on-demand process.

```
apex.server.process ( "MY_PROCESS", {
        x01: "test",
        pageItems: "#P1_DEPTNO,#P1_EMPNO"
  },{
        success: function( pData )
            // do something here
        }
  } );
```

31.11.5 apex.server.url

This function returns the URL to issue a GET request to the current page.

Parameters

Table 31-64 apex.server.url

Name	Type	Optional/Required	Description
`pData`	Object	Optional	Object which can optionally be used to set additional values which are included into the URL. The special attribute `pageItems` which can be of type jQuery selector, jQuery or DOM object or array of item names identifies the page items which are included in the URL. You can also set additional parameters that the `apex.show` procedure provides (for example you can set the scalar parameters x01 - x10 and the arrays f01 - f20).
`pPage`	String	Optional	Page ID of the current page, which can be optionally used to set the page ID in the URL being returned.

Returns

Returns the string URL to issue the GET request.

Example

This example demonstrates a call to `apex.server.url`. Returns a URL to issue a GET request to the DELETE function which has been defined for this page, where the URL sets the scalar value `x01` to `test` (which can be accessed from PL/SQL using `apex_application.g_x01`) and will also set the page item's `P1_DEPTNO` and `P1_EMPNO` values in session state (using jQuery selector syntax).

```
apex.server.url ({
    p_request: "DELETE",
    x01: "test",
    pageItems: "#P1_DEPTNO,#P1_EMPNO" });
```

31.12 apex.storage namespace

The `apex.storage` namespace contains all functions related browser storage features such as cookies and session storage.

31.12.1 About local and session storage

Local storage and session storage, collectively known as web storage, are a browser feature that securely stores key value pairs associated with an origin (web site). The keys and values are strings. The amount of storage space for web storage is greater than that of cookies but it is not unlimited. Another advantage over cookies is that the key value pars are not transmitted with each request.

Both local storage and session storage use the same API to set, get, and remove name value pairs. The difference is that session storage goes away when the browser session ends and local storage is available even when the browser restarts. Keep in mind that the browser is free to limit or delete data stored in local storage at the user's request. Unlike data stored on the server local storage is not shared between browsers on different machines or even different browsers on the same machine.

Because APEX supports multiple applications, multiple workspaces and even instances of the same application running in multiple workspaces there can arise conflicts with using web storage because all the apps from a single APEX instance (which is a single origin or web site) share the same web storage space. The `apex.storage.getScopedLocalStorage` and `apex.storage.getScopedSessionStorage` solve this problem buy partitioning the storage into a scope based on application id an optionally additional information such as page id and region id. The scope is crated by using a prefix on all the storage keys. This avoids conflicts when different apps or different instances of the same app use the same keys but it is not a secure partition. Consider this carefully before storing sensitive information in web storage.

31.12.2 apex.storage.getCookie

Returns the value of cookie name (`pName`).

Parameters

Table 31-65 apex.storage.getCookie

Name	Type	Description
`pName`	String	The name of the cookie.

Returns

The string value of the cookie.

31.12.3 apex.storage.hasLocalStorageSupport

Returns `true` if the browser supports the local storage API and false otherwise. Most modern browsers support this feature but some allow the user to turn it off.

Parameter

None.

Returns

`true` if the browser supports the local storage API and `false` otherwise.

31.12.4 apex.storage.getScopedLocalStorage

Returns a thin wrapper around the `localStorage` object that scopes all keys to a prefix defined by the `pOptions` parameter. If `localStorage` is not supported the returned object can be used but has no effect so it is not necessary test for support using `apex.storage.hasLocalStorageSupport` before calling this function.

Parameter

Table 31-66 apex.storage.getScopedLocalStorage

Name	Type	Description
`pOptions`	Object	An object that defines the scope of the local storage. This defines the storage key prefix used by the returned `localStorage` wrapper object. Refer to the following `pOptions` properties table.

Table 31-67 pOptions properties

Name	Type	Description
`prefix`	String	A static prefix string to add to all keys. The default is an empty string.
`useAppId`	Boolean	If `true` the application id will be included in the key The default is `true`.
`usePageId`	Boolean	If `true` the application page id will be included in the key. The default is `false`.
`regionId`	String	An additional string to identify a region or other part of a page.

Returns

A `localStorage` wrapper object. This object has the same properties and functions as the browser Storage interface.

Table 31-68 localStorage wrapper

Property	Description
`prefix`	APEX specific property. The prefix for this scoped storage object.
`length`	The number of items in the scoped storage object.

Table 31-68 (Cont.) localStorage wrapper

Property	Description
`key( n )`	Returns the nth key in the scoped storage object.
`getItem( key )`	Returns the value for the given key.
`setItem( key, data )`	Sets the value of the given key to data.
`removeItem( key )`	Removes the given key.
`clear:()`	Removes all keys from the scoped storage object.
`sync()`	APEX specific function. Use to ensure the length property is correct if keys may have been added or removed by means external to this object

Example

The following example creates a local storage object that scopes all the keys using a prefix "`Acme`" and the application id. It then sets and gets an item called "`setting1`".

```
var myStorage = apex.storage.getScopedLocalStorage( {
    prefix: "Acme",
    useAppId: true
});
myStorage.setItem( "setting1", "on" );
setting = myStorage.getItem( "setting1" ); // returns "on"
```

31.12.5 apex.storage.getScopedSessionStorage

Returns a thin wrapper around the `sessionStorage` object that scopes all keys to a prefix defined by the `pOptions` parameter. If sessionStorage is not supported the returned object can be used but has no effect so it is not necessary test for support using apex.storage.hasSessionStorageSupport before calling this function.

Parameter

Table 31-69 apex.storage.getScopedSessionStorage

Name	Type	Description
`pOptions`	Object	An object that defines the scope of the session storage. This defines the storage key prefix used by the returned `sessionStorage` wrapper object. Refer to the following `pOptions` properties table.

Table 31-70 pOptions properties

Name	Type	Description
`prefix`	String	A static prefix string to add to all keys. The default is an empty string.
`useAppId`	Boolean	If `true` the application id will be included in the key The default is `true`.
`usePageId`	Boolean	If `true` the application page id will be included in the key. The default is `false`.
`regionId`	String	An additional string to identify a region or other part of a page.

Returns

A `sessionStorage` wrapper object. This object has the same properties and functions as the browser Storage interface.

Table 31-71 sessionStorage

Property	Description
`prefix`	APEX specific property. The prefix for this scoped storage object.
`length`	The number of items in the scoped storage object.
`key( n )`	Returns the nth key in the scoped storage object.
`getItem( key )`	Returns the value for the given key.
`setItem( key, data )`	Sets the value of the given key to data.
`removeItem( key )`	Removes the given key.
`clear:()`	Removes all keys from the scoped storage object.
`sync()`	APEX specific function. Use to ensure the length property is correct if keys may have been added or removed by means external to this object

Example

The following example creates a session storage object that scopes all the keys using a prefix "`Acme`" and the application id. It then sets and gets an item called "`setting1`".

```
var myStorage = apex.storage.getScopedSessionStorage( {
    prefix: "Acme", useAppId: true
});
myStorage.setItem( "setting1", "on" );
setting = myStorage.getItem( "setting1" ); // returns "on"
```

31.12.6 apex.storage.hasSessionStorageSupport

Returns `true` if the browser supports the session storage API and false otherwise. Most modern browsers support this feature but some allow the user to turn it off.

Parameter

None.

Returns

`true` if the browser supports the session storage API and `false` otherwise.

31.12.7 apex.storage.setCookie

Sets a cookie (`pName`) to a specified value (`pValue`).

Parameters

Table 31-72 apex.storage.setCookie

Name	Type	Description
`pName`	String	The name of the cookie to set.
`pValue`	String	The value to set the cookie to.

31.13 apex.util namespace

`apex.util namespace` contains general utility functions of Oracle Application Express.

31.13.1 apex.util.escapeCSS

Returns string `pValue` with any CSS meta-characters are escaped. Use this function when the value is used as in a CSS selector. Whenever possible constrain the value so that it cannot contain CSS meta-characters making it unnecessary to use this function.

Parameters

Table 31-73 apex.util.escapeCSS

Name	Type	Description
pValue	String	The string that may contain CSS meta-characters to be escaped.

Returns

The escaped string.

Example

This example escapes an element id that contains a (.) period character so that it finds the element with id = "my.id". Without using this function the selector would have a completely different meaning.

```
apex.jQuery( "#" + apex.util.escapeCSS( "my.id" ) );
```

31.13.2 apex.util.escapeHTML

Returns string `pValue` with any special HTML characters `(&<>"'/)` escaped to prevent cross site scripting (XSS) attacks.

Note:

This function should always be used when inserting untrusted data into the DOM.

Parameters

Table 31-74 apex.util.escapeHTML

Name	Type	Description
pValue	String	The string that may contain special HTML characters to be escaped.

Returns

The escaped string.

Example

This example appends text to a DOM element where the text comes from a page item called `P1_UNTRUSTED_NAME`. Data entered by the user cannot be trusted to not contain malicious markup.

```
apex.jQuery( "#show_user" ).append( apex.util.escapeHTML( $v("P1_UNTRUSTED_NAME") ) )
;
```

31.13.3 apex.util.showSpinner

Function that renders a spinning alert to show the user processing is taking place. Note that the alert is defined as an ARIA alert so that assistive technologies such as screen readers are alerted to the processing status.

Parameters

Table 31-75 util.showSpinner

Name	Type	Description
`pContainer`	Object	Optional jQuery selector, jQuery, or DOM object identifying the container within which you want to center the spinner. If not passed, the spinner will be centered on the whole page. The default is `$("body")`.
`pOptions`	Object	Optional object with the following options: - "alert" Alert text visually hidden, but available to assistive technologies. Defaults to Processing. The default is `Processing`.

Returns

jQuery object for the spinner.

Example

To show the spinner:

```
var lSpinner$ = apex.util.showSpinner( $( "#container_id" ) );
```

To remove the spinner:

```
lSpinner$.remove();
```

31.13.4 apex.util.stripHTML

Return string `pText` with all HTML tags removed.

Parameters

Table 31-76 apex.util.stripHTML

Name	Type	Description
`pText`	String	A string that may contain HTML markup that you want removed.

Returns

The input string with all HTML tags removed.

Example

This example removes HTML tags from a text string.

```
apex.util.stripHTML("Please <a href='www.example.com/ad'>click here</a>")
// result “Please click here”
```

31.13.5 apex.util.applyTemplate

This function applies data to a template. It processes the template string given in `pTemplate` by substituting values according to the options in `pOptions`. The template supports Application Express server style placeholder and item substitution syntax.

This function is intended to process Application Express style templates in the browser. However it doesn't have access to all the data that the server has. When substituting page items and column items it uses the current value stored in the browser not what is in session state on the server. It does not support the old non-exact substitutions (with no trailing dot e.g. &ITEM). It does not support the old column reference syntax that uses `#COLUMN_NAME#`. It cannot do `prepare_url` (this must be done on the server). Using a template to insert JavaScript into the DOM is not supported. The template after processing will have script tags removed.

The format of a template string is any text intermixed with any number of replacement tokens. Two kinds of replacement tokens are supported: placeholders and substitutions. Placeholders are processed first.

Placeholder syntax is:

```
#<placeholder-name>#
```

The <placeholder-name> is an uppercase alpha numeric plus "_", and "$" string that must be a property name in option object placeholders that gets replaced with the property value. Any placeholder tokens that don't match anything in the placeholders object are left as is (searching for the next placeholder begins with the trailing # character).

Substitution syntax is (any of):

```
&<item-name>.
&<item-name>!<escape-filter>.
&"<quoted-item-name>".
&"<quoted-item-name>"!<escape-filter>.
```

The <item-name> is an uppercase alpha numeric plus "_", "$", and "#" string. The <quoted-item-name> is a string of any characters except "&", carriage return, line feed, and double quote. In both cases the item name is the name of a page item (unless option `includePageItems` is false), a column item (if model and record options are given), a built-in substitution (unless option `includeBuiltinSubstitutions` is false), or an extra substitution if option `extraSubstitutions` is given. If no replacement for a substitution can be found it is replaced with an empty string. When substituting a column item the given record of the given model is used to find a matching column name. If not found and if the model has a parent model then the parent model's columns are checked. This continues as long as there is a parent model. The order to resolve an item name is: page item, column item, column item from ancestor models, built-in substitutions, and finally extra substitutions. Column items support the `_LABEL` suffix to access the defined column label. For example if there is a column item named `NOTE` the substitution `&NOTE_LABEL.` will return the label string for column `NOTE`.

The built-in substitution names are:

- APP_ID

- APP_PAGE_ID
- APP_SESSION
- REQUEST
- DEBUG
- IMAGE_PREFIX

The escape-filter controls how the replacement value is escaped or filtered. It can be one of the following values:

- HTML the value will have HTML characters escaped.
- ATTR the value will be escaped for an HTML attribute context (currently the same as HTML)
- RAW does not change the value at all.
- STRIPHTML the value will have HTML tags removed and HTML characters escaped.

This will override any default escape filter set with option `defaultEscapeFilter` or from the column definition escape property.

Parameters

Table 31-77 defaultEscapeFilter

Name	Type	Description
`pTemplate`	String	A template string with the format described above.
`pOptions`	Object	An object that specifies how the template is to be processed. Refer to the following `pOptions` properties table.

Table 31-78 pOptions properties

Name	Description
`placeholders`	An object map of placeholder names to values. The default is`null`.
`defaultEscapeFilter`	One of the above escape-filter values. The default is HTML. This is the escaping/ filtering that is done if the substitution token doesn't specify an escape-filter. If a model column definition has an escape property then it will override the default escaping.
`includePageItems`	If `true` the current value of page items are substituted. The default is `true`.

Table 31-78 (Cont.) pOptions properties

Name	Description
`model`	The Application Express model used to get column item values. The default is `null`.
`record`	The record in the model to get column item values from. Option model must also be provided. The default is `null`.
`extraSubstitutions`	An object map of extra substitutions. The default is `null`.
`includeBuiltinSubstitutions`	If `true` built-in substitutions such as `APP_ID` are done. The default is `true`.

Returns

A string that is the template with all the replacement tokens substituted.

Examples

This example inserts an image tag where the path to the image comes from the built-in `IMAGE_PREFIX` substitution and a page item called `P1_PROFILE_IMAGE_FILE`.

```
apex.jQuery("#photo").html(
    apex.util.applyTemplate(
        "<img src='&IMAGE_PREFIX.people/&P1_PROFILE_IMAGE_FILE.'>" ) );
```

This example inserts a div with a message where the message text comes from a placeholder called `MESSAGE`.

```
var options = { placeholders: { MESSAGE: "All is well" } };
apex.jQuery("#notification").html( apex.util.applyTemplate( "<div>#MESSAGE#</div>",
options ) );
```

31.13.6 About apex.util.delayLinger

The `delayLinger` singleton solves the problem of flashing progress indicators (such as spinners). For processes such as an Ajax request (and subsequent user interface update) that may take a while it is important to let the user know that something is happening. The problem is that if an async process is quick there is no need for a progress indicator. The user experiences the user interface update as instantaneous. Showing and hiding a progress indicator around an async process that lasts a very short time causes a flash of content that the user may not have time to fully perceive. At best this can be a distraction and at worse the user wonders if something is wrong or if they missed something important. Simply delaying the progress indicator doesn't solve the problem because the process could finish a short time after the indicator is shown. The indicator must be shown for at least a short but perceivable amount of time even if the request is already finished.

You can use this object to help manage the duration of a progress indication such as `apex.util.showSpinner` or with any other progress implementation. Many of the Oracle Application Express asynchronous functions such as the ones in the `apex.server namespace` already use `delayLinger` internally so you only need this API for your own custom long running asynchronous processing.

31.13.7 apex.util.delayLinger.start

Call this function when a potentially long running async process starts. For each call to start with a given `pScopeName`, you must make a corresponding call to finish with the same `pScopeName`. Calls with different `pScopeName` arguments will not interfere with each other.

Multiple calls to start for the same `pScopeName` before any calls to finish is allowed but only the `pAction` from the first call is called at most once.

Parameters

Table 31-79 apex.util.delayLinger.start Function Parameters

Name	Type	Description
`pScopeName`	String	Unique name for each unique progress indicator.
`pAction`	Function	Function called to display the progress indicator.

31.13.8 apex.util.delayLinger.finish

Call this function when a potentially long running async process finishes. For each call to start with a given `pScopeName`, you must make a corresponding call to finish with the same `pScopeName`. The `pAction` is called exactly once if and only if the corresponding start pAction was called. If there are multiple calls to finish, the `pAction` from the last one is called.

Parameters

Table 31-80 apex.util.delayLinger.finishFunction Parameters

Name	Type	Description
`pScopeName`	String	Unique name for each unique progress indicator
`pAction`	Function	Function to call to display the progress indicator

Example

```
var lSpinner$, lPromise;
lPromise = doLongProcess();
util.delayLinger.start( "main", function() {
    lSpinner$ = apex.util.showSpinnter( $( "#container_id" ) );
} );
lPromise.always( function() {
    util.delayLinger.finish( "main", function() {
        lSpinner$.remove();
    } );
} );
```

31.14 apex.widget namespace

The `apex.widget` namespace stores all the general purpose widget related functions of Oracle Application Express.

31.14.1 apex.widget.initPageItem

This function provides backward compatibility for existing item plug-ins. New plug-ins should use `apex.item.create`.

See Also:

"apex.item.create (page 31-19)"

31.15 Events

Application Express specific events are discussed in the following section. In most cases you will use Dynamic Actions to respond to browser and APEX specific events. This section is for advanced JavaScript programmers that want to use these events directly from their JavaScript code.

31.15.1 apexafterclosedialog

This event is triggered when an Application Express modal dialog page is closed by either the Dynamic Action Close Dialog action or the Close Dialog process. This is equivalent to the Dialog Closed Dynamic Action event. This event is triggered on the element that opened the dialog.

For buttons it is the button element. For links to dialog pages in lists it is the list region element. For lists used in global top or side navigation or in any other case where the triggering element cannot be determined the event is triggered on the document `apex.gPageContext$`. The event handler receives an object argument with these properties.

Table 31-81 apexafterclosedialog

Property	Type	Description
`dialogPageId`	String	The page number of the dialog page that closed.
*	String	For each page item listed in the Close Dialog process or dynamic action setting Items to Return there is a property with the same name as the item. The value is the value of the item.

Note:

This event is triggered in the parent or calling page not in the modal dialog page. If you want to know when the dialog is closed regardless of how it is closed use the jQuery UI dialogclose event.

Example

This example refreshes the region with static id `emp` when any modal dialog page closes.

```
apex.gPageContext$.on( "apexafterclosedialog", function( event, data ) {
    apex.region( "emp" ).refresh();
});
```

31.15.2 apexafterrefresh

This event is triggered by a number of page or column items just after they are refreshed with new content or data from the server. It is equivalent to the Dynamic Action event After Refresh. Specifically any item that supports the Cascading LOV Parent Item(s) attribute should trigger this event. This event can also be triggered by the `apex.server.plugin` and `apex.server.process` APIs if the `refreshObject` option is provided. The event is triggered on the item element or the element given by the `refreshObject`. The event handler receives the data given in `refreshObjectData` if any.

Example

This example disables the button with static id `B1` while any refresh is in progress.

```
apex.jQuery( "body" ).on( "apexbeforerefresh", function() {
    apex.jQuery( "#B1" ).prop( "disabled", true);
}).on( "apexafterrefresh", function() {
    apex.jQuery( "#B1" ).prop( "disabled", false);
});
```

See Also:

"apex.server namespace (page 31-61)"

31.15.3 apexbeforerefresh

This event is triggered by a number of page or column items just before they are refreshed with new content or data from the server. It is equivalent to the Dynamic Action event Before Refresh. Specifically any item that supports the Cascading LOV Parent Item(s) attribute should trigger this event. This event can also be triggered by the `apex.server.plugin` and `apex.server.process` APIs if the `refreshObject` option is provided. The event is triggered on the item element or the element given by the `refreshObject`. The event handler receives the data given in `refreshObjectData` if any.

Example

This example disables the button with static id `B1` while any refresh is in progress.

```
apex.jQuery( "body" ).on( "apexbeforerefresh", function() {
    apex.jQuery( "#B1" ).prop( "disabled", true);
}).on( "apexafterrefresh", function() {
    apex.jQuery( "#B1" ).prop( "disabled", false);
});
```

See Also:

"apex.server namespace (page 31-61)"

31.15.4 apexbeforepagesubmit

This event is triggered when the page is submitted with `apex.submit` or `apex.confirm`. This includes buttons with action Submit Page and Dynamic Action Submit Page action. It is equivalent to the Dynamic Action event Before Page Submit. It is triggered before the page is validated. It is triggered on `apex.gPageContext$`, which is the document for Desktop UI pages and the page div for jQuery Mobile UI pages. This event can be canceled by a Dynamic Action Confirm or Cancel Event action so you cannot rely on the page actually being submitted. If you need code to run just before the page is actually submitted see the `apexpagesubmit` event.

The event handler should not do any long running or asynchronous processing. Specifically it should not make a synchronous or asynchronous Ajax request. The event handler receives a string argument that is the request value.

Example

This example performs an extra validation on page item `P1_CHECK_ME`. For this to work the Submit button Execute Validations attribute must be `Yes` and the application compatibility mode must be greater than or equal to 5.1 or the validate option to `apex.submit` or `apex.confirm` must be true.

```
apex.jQuery( apex.gPageContext$ ).on( "apexbeforepagesubmit", function() {
    var item = apex.item("P1_CHECK_ME" ),
        value = item.getValue();
    if ( value !== "valid" ) { // replace with desired constraint check
        item.node.setCustomValidity( "Text field needs to be valid" );
    }else {
        item.node.setCustomValidity( "" );
    }
});
```

See Also:

"apexpagesubmit (page 31-89)"

31.15.5 apexbeginrecordedit

This event is triggered when a region that supports column items such as Interactive Grid begins editing a record. In general it is simpler to use a Dynamic Action on a column item and set attribute Fire on Initialization to `Yes`. This is equivalent to the Interactive Grid Row Initialization Dynamic Action event.

It is triggered on the widget that is doing the editing. It is triggered after all the column items have been initialized. The event handler receives an object argument with these properties.

Table 31-82 apexbeginrecordedit

Property	Type	Description
`recordId`	String	The value of the primary key column. If there is more than one primary key then this is a serialized JSON array of primary key values.
`model`	Model	The model used by the region widget.
`record`	Array | Object	The current record being edited. The record belongs to the model given in property model and should only be modified using the model API.

Note:

using the model or record requires using the undocumented `apex.model API`.

Example

This example responds to the `apexbeginrecordedit` event of an Interactive Grid, which edits the EMP table. The region was given a static id of `emp`. The SAL column was given a static id of `c_sal`.

```
apex.region( "emp" ).widget().on( "apexbeginrecordedit", function( event, data ) {
    var salary = apex.item( "c_sal" ).getValue();
     // use data.recordId or access any of the column items using apex.item API });
```

31.15.6 apexendrecordedit

This event is triggered when a region that supports column items such as Interactive Grid ends editing a record.

It is triggered on the widget that is doing the editing. It is triggered after all the model has been updated with any column item changes and the column items have been validated. The event handler receives an object argument with these properties.

Table 31-83 apexendrecordedit

Property	Type	Description
`recordId`	String	The value of the primary key column. If there is more than one primary key then this is a serialized JSON array of primary key values.
`model`	Model	The model used by the region widget.
`record`	Array | Object	The current record being edited. The record belongs to the model given in property model and should only be modified using the model API.

Note:

using the model or record requires using the undocumented `apex.model API`.

Example

This example responds to the `apexendrecordedit` event of an Interactive Grid, which edits the EMP table. The region was given a static id of `emp`. The SAL column was given a static id of `c_sal`.

```
apex.region( "emp" ).widget().on( "apexendrecordedit", function( event, data ) {
    var salary = apex.item( "c_sal" ).getValue();
     // use data.recordId or access any of the column items using apex.item API });
```

31.15.7 apexpagesubmit

This event is triggered when the page is submitted with `apex.submit` or `apex.confirm`. This includes buttons with action Submit Page and Dynamic Action Submit Page action. It is triggered after the page is validated. It is triggered on `apex.gPageContext$`, which is the document for Desktop UI pages and the page div for jQuery Mobile UI pages. This event is the last chance to set or modify page items before the page is submitted.

The event handler should not do any long running or asynchronous processing. Specifically it should not make a synchronous or asynchronous Ajax request. The event handler receives a string argument that is the request value.

Example

This example makes the page item `P1_VALUE` upper case before the page is submitted.

```
apex.jQuery( apex.gPageContext$ ).on( "apexpagesubmit", function() {
    var item = apex.item("P1_VALUE");
    item.setValue( item.getValue().toUpperCase());
});
```

31.15.8 apexwindowresized

This event is triggered on the window a couple hundred milliseconds after the window stops resizing. Listen for this event to adjust or resize page content after the window is done resizing. In some cases this is a better alternative to the window resize event, which is triggered many times as the window is being resized, because it is triggered just once after the window stops resizing.

Example

This example responds to the `apexwindowresized` event and updates page content based on the new height and width.

```
apex.jQuery( window ).on( "apexwindowresized", function( event ) {
    var window$ = apex.jQuery( this ),
          height = window$.height(),
          width = window$.width();
    // update page content based on new window height and width
});
```

31.16 Non-namespace JavaScript APIs

All the miscellaneous, non-namespace APIs of Oracle Application Express, including shortcuts to highly used functions are discussed in the following:

31.16.1 $x

Given a DOM node or string ID (pNd), this function returns a DOM node if the element is on the page, or returns `false` if it is not.

Parameters

```
pNd (DOM Node | string ID)
```

Returns

```
(DOM Node | false)
```

31.16.2 $v

Given a DOM node or string ID (pNd), this function returns the value of an Application Express item in the same format as it would be posted.

Parameters

```
pNd (DOM Node | string ID)
```

31.16.3 $v2

Given a DOM node or string ID (pNd), this function returns the value of an Application Express item as a string or an array. If the page item type can contain multiple values like a shuttle, checkboxes or a multi select list an array is returned, otherwise a string.

Parameters

```
pNd (DOM Node | string ID)
```

Returns

```
(string|array)
```

31.16.4 $s

Given a DOM node or string ID (`pNd`), this function sets the Application Express item value taking into account the item type. The `pDisplayValue` is optional. If used for a page item of type `Popup LOV` where the attribute `"Input Field" = "Not Enterable, Show Display Value and Store Return Value"`, it sets the `"Input Field"`. The value of `pValue` is stored in the hidden return field. The `pSuppressChangeEvent` parameter is optional. Passing either `false` or not passing this parameter value results in a change event firing for the item being set. Pass `true` to prevent the change event from firing for the item being set.

Parameters

```
pNd (DOM Node | string ID)
pValue  (String | Array)
pDisplayValue(String)
pSuppressChangeEvent(Boolean)
```

31.16.5 $u_Narray

Given a DOM node or string ID or an array (pNd), this function returns a single value, if an pNd is an array but only has one element the value of that element is returned otherwise the array is returned. Used for creating DOM based functionality that can accept a single or multiple DOM nodes.

Parameters

Array or first value

Returns

```
Array (DOM Node | string ID | Array)
```

31.16.6 $u_Carray

Given a DOM node or string ID or an array (pNd), this function returns an array. Used for creating DOM based functionality that can accept a single or multiple DOM nodes.

Parameters

```
Array
```

Returns

```
pNd (DOM Node | string ID | Array)
```

31.16.7 $nvl

If `pTest` is empty or `false` return `pDefault` otherwise return `pTest`.

Parameters

```
pTest  (String | Array)
pDefault (String | Array)
```

Returns

```
(string | Array)
```

31.16.8 $x_Style

Sets a specific style property (`pStyle`) to given value (`pString`) of a DOM node or DOM node Array (`pNd`).

Parameters

```
pNd (DOM node | string ID | DOM node Array )
pStyle (String)
pString (String)
```

Returns

```
(DOM node | DOM Array)
```

31.16.9 $x_Hide

Hides a DOM node or array of DOM nodes (`pNd`). This also takes into consideration which type of Application Express item is being hidden.

Parameters

```
pNd (DOM node | string ID | DOM node Array )
```

Returns

```
(DOM node | Array)
```

31.16.10 $x_Show

Shows a DOM node or array of DOM nodes (`pNd`). This also takes into consideration which type of Application Express item is being hidden.

Parameters

```
pNd (DOM node | string ID | DOM node Array )
```

Returns

```
(DOM node | Array)
```

31.16.11 $x_Toggle

Toggles a DOM node or array of DOM nodes (pNd).

Parameters

```
pNd (DOM node | string ID | Array)
```

Returns

```
(DOM node | Array)
```

31.16.12 $x_Remove

Removes a DOM node or array of DOM nodes.

Parameters

```
pNd (DOM node | string ID | DOM node Array)
```

Returns

```
(DOM Node | Array)
```

31.16.13 $x_Value

Sets the value (`pValue`) of a DOM node or array of DOM nodes (`pNd`).

Parameters

```
pNd (DOM node | string ID | DOM node Array)
pValue (String)
```

Returns

Not applicable.

31.16.14 $x_UpTill

Starting from a DOM node (`pNd`), this function cascades up the DOM tree until the tag of node name (`pToTag`) is found. If the optional `pToClass` is present, the ancestor node must have a node name that equals `pToTag` and the class must equal `pToClass`.

Parameters

```
pNd  (DOM Node | string ID)
String (pToTag)
String (pToClass )
```

Returns

```
(DOM Node | false)
```

31.16.15 $x_ItemRow

Given DOM node or array of DOM nodes, this function (shows, hides, or toggles) the entire row that contains the DOM node or array of DOM nodes. This is most useful when using Page Items. This function only works in table layouts since it explicitly looks for a containing `tr` element.

Parameters

```
pNd (DOM Node | string ID | Dom node Array)
pFunc ['TOGGLE','SHOW','HIDE'] (String )
```

Returns

Not applicable.

31.16.16 $x_HideItemRow

Given a page item name, this function hides the entire row that holds the item. In most cases, this is the item and its label. This function only works in table layouts since it explicitly looks for a containing `tr` element.

Parameters

```
pNd (DOM Node | string ID | DON node Array)
```

Returns

Not applicable.

31.16.17 $x_ShowItemRow

Given a page item name, this function shows the entire row that holds the item. In most cases, this is the item and its label. This function only works in table layouts since it explicitly looks for a containing `tr` element.

Parameters

```
pNd (DOM node | string ID | DOM note Array)
```

Returns

Not applicable.

31.16.18 $x_ToggleItemRow

Given a page item name (pNd), this function toggles the entire row that holds the item. In most cases, this is the item and its label. This function only works in table layouts since it explicitly looks for a containing `tr` element.

Parameters

```
pNd (DOM node | string ID | DOM node ray)
```

Returns

Not applicable.

31.16.19 $x_HideAllExcept

Hides all DOM nodes referenced in `pNdArray` and then shows the DOM node referenced by `pNd`. This is most useful when `pNd` is also a node in `pNdArray`.

Parameters

```
pNd (DOM node | string ID | DOM node Array)
pNdArray (DOM node | String | Array)
```

Returns

```
(DOM node | DOM Array)
```

31.16.20 $x_HideSiblings

Hides all sibling nodes of given `pNd`.

Parameters

```
pNd (DOM node | string ID )
```

Returns

```
(DOM node)
```

31.16.21 $x_ShowSiblings

Shows all sibling DOM nodes of given DOM nodes (`pNd`).

Parameters

```
pNd (DOM node | string ID )
```

Returns

```
(DOM node)
```

31.16.22 $x_Class

Sets a DOM node or array of DOM nodes to a single class name.

Parameters

```
pNd (DOM node | string ID | DOM node Array)
pClass (String)
```

Returns

Not applicable.

31.16.23 $x_SetSiblingsClass

Sets the class (`pClass`) of all DOM node siblings of a node (`pNd`). If `pNdClass` is not null the class of `pNd` is set to `pNdClass`.

Parameters

```
pNd (DOM Nnde | string ID)
pClass (String)
pThisClass (String)
```

Returns

```
(DOM node | false)
```

31.16.24 $x_ByClass

Returns an array of DOM nodes by a given class name (`pClass`). If the `pNd` parameter is provided, then the returned elements are all children of that DOM node. Including the `pTag` parameter further narrows the list to just return nodes of that tag type.

Parameters

```
pClass (String)
pNd  (DOM node | string ID)
pTag (String)
```

Returns

```
(Array)
```

31.16.25 $x_ShowAllByClass

Show all the DOM node children of a DOM node (`pNd`) that have a specific class (`pClass`) and tag (`pTag`).

Parameters

```
pNd (DOM node | string ID)
pClass (String)
pTag (String)
```

Returns

Not applicable.

31.16.26 $x_ShowChildren

Show all DOM node children of a DOM node (`pNd`).

Parameters

```
pNd (DOM node | string ID)
```

Returns

Not applicable.

31.16.27 $x_HideChildren

Hide all DOM node children of a DOM node (`pNd`).

Parameters

```
pNd (DOM node | string ID)
```

Returns

Not applicable.

31.16.28 $x_disableItem

Disables or enables an item or array of items based on (`pTest`).

Parameters

```
pNd (DOM node | string ID | DOM node array)
a (true | false)
```

Returns

Not applicable.

31.16.29 $f_get_emptys

Checks an item or an array of items to see if any are empty, set the class of all items that are empty to `pClassFail`, set the class of all items that are not empty to `pClass`.

Parameters

```
pNd (DOM node | string ID | DOM node Array)
Sting (pClassFail)
Sting (pClass)
```

Returns

```
false, Array  Array of all items that are empty (false | Array)
```

31.16.30 $v_Array

Returns an item value as an array. Useful for multiselects and checkboxes.

Parameters

```
pId (DOM Node | string ID)
```

Returns

```
(Array)
```

31.16.31 $f_ReturnChecked

Returns an item value as an array. Useful for radio items and check boxes.

Parameters

```
pId (DOM node | string ID)
```

Returns

```
(Array)
```

31.16.32 $d_ClearAndHide

Clears the content of an DOM node or array of DOM nodes and hides them.

Parameters

```
pNd (DOM node | string ID | DOM node array)
```

Returns

Not applicable.

31.16.33 $f_SelectedOptions

Returns the DOM nodes of the selected options of a select item (`pNd`).

Parameters

```
pNd (DOM node | string ID)
```

Returns

```
(DOM Array)
```

31.16.34 $f_SelectValue

Returns the values of the selected options of a select item (`pNd`).

Parameters

```
pNd (DOM node | string ID)
```

Returns

```
(DOM Array | String)
```

31.16.35 $u_ArrayToString

Given an array (`pArray`) return a string with the values of the array delimited with a given delimiter character (`pDelim`).

Parameters

```
pArray (pArray)
pDelim (String)
```

Returns

Not applicable.

31.16.36 $x_CheckImageSrc

Checks an image (`pId`) `source` attribute for a substring (`pSearch`). The function returns `true` if a substring (`pSearch`) is found. It returns `false` if a substring (`pSearch`) is not found.

Parameters

```
pId (DOM Node | String)
pSearch (pSearch)
```

Returns

```
(true | false)
```

31.16.37 $v_CheckValueAgainst

Checks an page item's (`pThis`) value against a set of values (`pValue`). This function returns `true` if any value matches.

Parameters

```
pThis (DOM node | string ID)
pValue (Number | String | Array)
```

Returns

```
(true | false)
```

31.16.38 $f_Hide_On_Value_Item

Checks page item's (`pThis`) value against a value (`pValue`). If it matches, a DOM node (`pThat`) is set to hidden. If it does not match, then the DOM node (`pThat`) is set to visible.

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

Returns

```
(true | false)
```

31.16.39 $f_Show_On_Value_Item

Checks page item's (`pThis`) value against a value (`pValue`). If it matches, a DOM node (`pThat`) is set to visible. If it does not match, then the DOM node (`pThat`) is set to hidden.

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

Returns

```
(true | false)
```

31.16.40 $f_Hide_On_Value_Item_Row

Checks the value (`pValue`) of an item (`pThis`). If it matches, this function hides the table row that holds (`pThat`). If it does not match, then the table row is shown.

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

Returns

```
(true | false)
```

31.16.41 $f_Show_On_Value_Item_Row

Checks the value (`pValue`) of an item (`pThis`). If it matches, this function shows the table row that holds (`pThat`). If it does not match, then the table row is hidden.

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

Returns

```
(true | false)
```

31.16.42 $f_DisableOnValue

Checks the value (`pValue`) of an item (`pThis`). If it matches, this function disables the item or array of items (`pThat`). If it does not match, then the item is enabled.

Parameters

```
pThis (DOM node | string ID)
pValue (String)
pThat  (DOM node | string ID | DOM node Array )
```

Returns

```
(true | false)
```

31.16.43 $x_ClassByClass

Sets a class attribute of an array of nodes that are selected by class.

Parameters

```
pNd (DOM node | string ID)
pClass (String)
pTag (String)
pClass2 (String)
```

Returns

```
(DOM node | DOM node Array)
```

31.16.44 $f_ValuesToArray

Collects the values of form items contained within DOM node (`pThis`) of class attribute (`pClass`) and nodeName (`pTag`) and returns an array.

Parameters

```
pThis (DOM node | string ID)
pCLass (String)
pTag (String)
```

Returns

Not applicable.

31.16.45 $x_FormItems

Returns all form input items contained in a DOM node (`pThis`) of a certain type (`pType`).

Parameters

```
pNd (DOM node | string ID)
pType (String)
```

Returns

```
DOM node Array
```

31.16.46 $f_CheckAll

Check or uncheck (`pCheck`) all check boxes contained within a DOM node (`pThis`). If an array of checkboxes DOM nodes (`pArray`) is provided, use that array for affected check boxes.

Parameters

```
pThis (DOM node | string ID)
pCheck (true | fales)
pArray (DOM node array)
```

Returns

Not applicable.

31.16.47 $f_CheckFirstColumn

This function sets all checkboxes located in the first column of a table based on the checked state of the calling check box (`pNd`), useful for tabular forms.

Parameters

```
pNd (DOM node | String)
```

Returns

```
DOM node Array
```

31.16.48 $x_ToggleWithImage

Given an image element (`pThis`) and a DOM node (`pNd`), this function toggles the display of the DOM node (`pNd`). The src attribute of the image element (`pThis`) is

rewritten. The image src has any plus substrings replaced with minus substrings or minus substrings are replaced with plus substrings.

Parameters

```
pThis (DOM Node | string ID)
pNd (DOM Nnde | string iD | DOM node Array)
```

Returns

```
(DOM Node)
```

31.16.49 $x_SwitchImageSrc

Checks an image (`pId`) src attribute for a substring (`pSearch`). If a substring is found, this function replaces the image entire src attribute with (`pReplace`).

Parameters

```
pNd (DOM node | string ID)
pSearch (String)
pReplace (String)
```

Returns

```
(DOM node | false)
```

31.16.50 $x_CheckImageSrc

Checks an image (`pNd`) source attribute for a substring (`pSearch`). The function returns `true` if a substring (`pSearch`) is found. It returns `false` if a substring (`pSearch`) is not found.

Parameters

```
pNd  (DOM node | string ID)
pSearch (String)
```

Returns

```
(true | fales)
```

31.16.51 $u_SubString

Returns a `true` or `false` if a string (`pText`) contains a substring (`pMatch`).

Parameters

```
pText (String)
pMatch (String)
```

Returns

```
(true | false)
```

31.16.52 html_RemoveAllChildren

Use DOM methods to remove all DOM children of DOM node (`pND`).

Parameters

```
pNd (DOM node | string ID)
```

Returns

Not applicable.

31.16.53 html_SetSelectValue

Sets the value (`pValue`) of a select item (`pId`). If the value is not found, this functions selects the first option (usually the `NULL` selection).

Parameters

```
pId (DOM node | String)
pValue (String)
```

Returns

Not applicable.

31.16.54 addLoadEvent

Adds an onload function (`func`) without overwriting any previously specified onload functions.

Parameters

```
pFunction (Javascript Function)
```

Returns

Not applicable.

31.16.55 $f_Swap

Swaps the form values of two form elements (`pThis`,`pThat`).

Parameters

```
pThis (DOM Node | String)
pThat (DOM Node | String)
```

Returns

Not applicable.

31.16.56 $f_SetValueSequence

Sets array of form item (`pArray`) to sequential number in multiples of (`pMultiple`).

Parameters

```
pArray (Array)
pMultiple (Number)
```

Returns

Not applicable.

31.16.57 $dom_AddTag

Inserts the html element (`pTag`) as a child node of a DOM node (`pThis`) with the innerHTML set to (`pText`).

Parameters

```
pThis (DOM node | string ID )
pTag (String)
pText (String)
```

Returns

```
DOM node
```

31.16.58 $tr_AddTD

Appends a table cell to a table row (`pThis`). And sets the content to (`pText`).

Parameters

```
pThis (DOM node | string ID)
pText (String)
```

Returns

```
(DOM node)
```

31.16.59 $tr_AddTH

Appends a table cell to a table row (`pThis`). And sets the content to (`pText`).

Parameters

```
pThis (DOM node | string ID)
pTest (String)
```

Returns

```
DOM node
```

31.16.60 $dom_AddInput

Inserts the html form input element (`pType`) as a child node of a DOM node (`pThis`) with an id (`pId`) and name (`pName`) value set to `pValue`.

Parameters

```
pThis (DOM node | string ID)
pType (String)
pId (String)
pName (String)
pValue (String)
```

Returns

```
(DOM node)
```

31.16.61 $dom_MakeParent

Takes a DOM node (`p_Node`) and makes it a child of DOM node (`p_Parent`) and then returns the DOM node (pNode).

Parameters

```
p_This (DOM node | string ID)
p_Parent (DOM node | string ID)
```

Returns

```
(DOM node)
```

31.16.62 $x_RowHighlight

Give an table row DOM element (`pThis`), this function sets the background of all table cells to a color (`pColor`). A global variable `gCurrentRow` is set to `pThis`.

Parameters

```
pThis (DOM node | String)
pColor(String)
```

Returns

Not applicable.

31.16.63 $x_RowHighlightOff

Give an table row Dom node (`pThis`), this function sets the background of all table cells to `NULL`.

Parameters

```
pThis (DOM Element | String)
```

Returns

Not applicable.

31.16.64 $v_Upper

Sets the value of a form item (pNd) to uppercase.

Parameters

```
pNd (DOM Node | String)
```

Returns

Not applicable.

31.16.65 $d_Find

Hides child nodes of a Dom node (`pThis`) where the child node's inner HTML matches any instance of `pString`. To narrow the child nodes searched by specifying a tag name (`pTag`) or a class name (`pClass`). Note that the child node is set to a block level element when set to visible.

Parameters

```
pThis (DOM node | String)
pString (String)
pTags (String
pClass (String)
```

Returns

Not applicable.

31.16.66 $f_First_field

Places the user focus on a form item (pNd). If pNd is not found then this function places focus on the first found user editable field.

Parameters

```
pNd
```

Returns

```
true (if successful)
```

31.17 Legacy JavaScript APIs

Work has commenced to reduce the overall size of JavaScript that is loaded by Oracle Application Express when rendering a page. JavaScript functions that are no longer served on every page are gradually being moved to a legacy JavaScript file, which can be found in `/i/libraries/apex/legacy.js`.

When developing applications, a developer has the option to either include, or not include the legacy JavaScript functions. This is achieved by using the Include Legacy JavaScript property on the User Interface Attributes page under the application's Shared Components.

Existing applications are migrated with this option enabled, for backward compatibility. To not include this legacy file, you need to go through the functions listed in the legacy file, and search your application and associated JavaScript files for any references to those files. If you are happy that there are no references to these functions, you can switch off including the legacy file and benefit from the slightly smaller library.

When developing new applications, the legacy file is included by default in all applications that use a Desktop User Interface Type. New applications that use a jQuery Mobile Smartphone User Interface Type do not include this file.

For both new and existing application development, Oracle recommends that you do not continue to use any of the functions in `legacy.js`, to reduce your dependency on this legacy JavaScript.

31.17.1 $v_PopupReturn [Deprecated]

Sets the value of the item in the parent window (`pThat`), with (`pValue`) and then closes the popup window.

Note:

This function is deprecated. Instead, use:

apex.navigation.popup (page 31-47)

For existing applications, the old function is still available, because of the application including the 'Legacy JavaScript' file (legacy.js). For details on how to control the inclusion of this file, see "About Database Applications" in *Oracle Application Express Application Builder User's Guide*.

Parameters

```
pValue (string)
pThat (DOM node | string ID)
```

Returns

Not applicable.

31.17.2 $v_IsEmpty [Deprecated]

Returns `true` or `false` if a form element is empty, this considers any whitespace including a space, a tab, a form-feed, as empty. This also considers any null value that has been specified on the item.

Note:

This function is deprecated. Instead, use:

isEmpty (page 31-15)

For existing applications, the old function is still available, because of the application including the 'Legacy JavaScript' file (legacy.js). For details on how to control the inclusion of this file, see "About Database Applications" in *Oracle Application Express App Builder User's Guide*.

Parameters

```
pThis (DOM Node | String)
```

Returns

```
[true | false]
```

31.17.3 submitEnter [Deprecated]

Submits a page when ENTER is pressed in a text field, setting the request value to the ID of a DOM node (`pNd`).

Usage is `onkeypress="submitEnter(this,event)"`

Note:

This function is deprecated. Instead, use:

```
apex.submit( { submitIfEnter : event })
```

See `apex.submit` for further details on how to use the '`submitIfEnter`' `pOptions` property.For existing applications, the old function is still available, because of the application including the 'Legacy JavaScript' file (legacy.js). For details on how to control the inclusion of this file, see "About Database Applications" in*Oracle Application Express App Builder User's Guide* .

Parameters

```
pNd (DOM node | String | Array)
```

Returns

Not applicable.

31.17.4 setReturn [Deprecated]

Sets DOM items in the global variables `returnInput` (`p_R`) and `returnDisplay` (`p_D`) for use in populating items from popups.

Note:

This function is deprecated and due to very limited value there is no alternative.

For existing applications, the old function is still available, because of the application including the 'Legacy JavaScript' file (legacy.js). For details on how to control the inclusion of this file, see "About Database Applications" in *Oracle Application Express App Builder User's Guide*.

Parameters

```
p_R
p_D
```

Returns

Not applicable.

31.17.5 GetCookie [Deprecated]

Returns the value of cookie name (`pName`).

Note:

This function is deprecated. Instead, use:

apex.storage.getCookie (page 31-74)

For existing applications, the old function is still available, because of the application including the 'Legacy JavaScript' file (legacy.js). For details on how to control the inclusion of this file, see "About Database Applications" in *Oracle Application Express App Builder User's Guide*.

Parameters

```
pName (String)
```

Returns

Not applicable.

31.17.6 SetCookie [Deprecated]

Sets a cookie (`pName`) to a specified value (`pValue`).

Note:

This function is deprecated. Instead, use:

apex.storage.setCookie (page 31-78)

For existing applications, the old function is still available, because of the application including the 'Legacy JavaScript' file (legacy.js). For details on how to control the inclusion of this file, see "About Database Applications" in *Oracle Application Express App Builder User's Guide*.

Parameters

```
pName (String)
pValue (String)
```

Returns

Not applicable.

32

Using REST Administration Interface API

The REST Administration API enables Oracle Application Express instance administrator to perform administrative functions in an Application Express instance over REST and HTTP protocols. This is particularly useful for machine-to-machine communication when `SQL*Net` connections are not possible, for instance in cloud environments. The REST Administration Interface enables administrators to automatically fetch usage metrics for an Oracle Application Express instance with a REST client.

Tip:

The REST Administration Interface requires Oracle REST Data Services (ORDS) release 3.0.5 or later.

See Also:

"Using the REST Administration Interface to View Usage Statistics" in *Oracle Application Express Administration Guide*

32.1 Authentication

REST clients must authenticate before accessing the administrative REST services. First, an Oracle Application Express instance administrator must log into the Oracle Application Express application and register a REST client.

When a client has been registered in Instance Administration, the dialog shows `Client ID` and `Client Secret`, with which the client can then perform authentication following the OAuth2 `Client Credentials` flow. A client first connects with a `Client ID` and a `Client Secret` as the credentials. Upon successful authentication, the server sends back the `OAuth Access Token`. Using this access token, the client can then access the administrative REST services.

HTTP Request Syntax Parameter

Table 32-1 HTTP Request Syntax

Parameter	Description
HTTP Method	`POST`

Table 32-1 (Cont.) HTTP Request Syntax

Parameter	Description
URL	`http://application-express-host:port/ords/apex_instance_admin_user/oauth/token`
Request Body	`grant_type=client_credentials`
HTTP Request Headers	`"Content-Type": "application/x-www-form-urlencoded"` `"Authorization": Client-ID:Client Secret in Base64-encoded form`

Returns

Returns a JSON object with the following structure upon successful authentication:

```
{
  "access_token": OAuth access token fot subsequent requests,
  "token_type":   "bearer",
  "expires_in":   lifetime of the OAuth token, in seconds; typically "3600"
}
```

If authentication is unsuccessful, the server responds with `HTTP-401:Unauthorized`.

Examples

In the following example`ClientID` stands for the Client ID and `ClientSecret` for the Client Secret.

Example 1

The example displays the following output when you execute command line utility `curl`:

```
   $ curl -i
          --user ClientId:ClientSecret
          --data "grant_type=client_credentials"
          http://application-express-host:port/ords/apex_instance_admin_user/oauth/
token

   HTTP/1.1 200 OK
   Content-Type: application/json
   Transfer-Encoding: chunked

   "access_token":"LfXJilIBdzj5JPRn4xb5QQ..","token_type":"bearer","expires_in":3600
```

Use a JSON parser to extract the value of the `access_token` attribute and use it in subsequent requests.

Example 2

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

```
  begin
       apex_web_service.oauth_authenticate(
           p_token_url =>     'http://application-express-host:port/ords/
apex_instance_admin_user/oauth/token',
           p_client_id =>     'ClientId',
           p_client_secret => 'ClientSecret'
       );
       dbms_output.put_line( 'The token is: ' ||
apex_web_service.oauth_get_last_token );
   end;
   /

   The token is: LfXJilIBdzj5JPRn4xb5QQ..
```

With the acquired OAuth Access Token, the administrative REST Services can be called.

See Also:

Oracle Application Express Administration Guide

32.2 Individual REST Services

Individual REST Services have the following services:

Fetch Instance-Level Statistics (page 32-3)

Fetch Workspace-Level Statistics (page 32-5)

Application-Level Statistics (page 32-8)

Instance Overview (page 32-10)

REST Service Version Information (page 32-12)

32.2.1 Fetch Instance-Level Statistics

This service returns usage statistics for the whole Oracle Application Express instance.

HTTP Request Syntax Parameter

Table 32-2 HTTP Request Syntax

Parameter	Description
HTTP Method	`GET`
URL	`http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/instance`
HTTP Request Headers	`"authorization": "Bearer: OAuth access token aquired with Authentication`

Returns

```
{
    "items": [
        {
            "log_day": "2016-09-15T00:00:00Z",
            "workspace_id": 1809074264671554,
            "workspace_name": "SOMEWORKSPACE",
            "workspace_link": {
                "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/workspace/someworkspace"
            },
            "application_id": 4750,
            "application_name": "Oracle APEX Packaged Applications",
            "application_link": {
```

Parameters

Filtering is possible for each of the JSON attributes by appending a query to the request URL as follows:

```
http://application-express-host.../latest/instance?q=query
```

Examples

Get All Instance Statistics

The example displays the following output when you execute command line utility `curl`:

```
  $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/stats/
latest/instance
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

Note:

The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
   select apex_web_service.make_rest_request(
            p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/instance'
            p_http_method => 'GET' )
   from dual;
```

The following is a JSON-Document response:

```
   {
       "items": [
           {
               "log_day": "2016-09-15T00:00:00Z",
               "workspace_id": 267781782378434879,
               "workspace_name": "SOMEWORKSPACE",
               "workspace_link": {
                   "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/workspace/someworkspace"
```

Get Instance Statistics Since August 1st, 2016

The example displays the following output when you execute command line utility `curl`:

```
  $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/stats/
latest/instance?q=%7B%22log_day%22:%7B%22$gt%22:%7B%22$date
%22:%222016-08-01T00:00:00Z%22%7D%7D%7D
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

> **Note:**
>
> The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
 select apex_web_service.make_rest_request(
           p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/instance?q=' ||
                            utl_url.escape('{"log_day": {"$gt": {"$date":
"2016-08-01T00:00:00Z"}}}'),
           p_http_method => 'GET',
           p_scheme =>      'OAUTH_CLIENT_CRED' )
   from dual;
```

The following is a JSON-Document response:

```
  {
       "items": [
           {
               "log_day": "2016-09-15T00:00:00Z",
               "workspace_id": 267781782378434879,
               "workspace_name": "SOMEWORKSPACE",
               "workspace_link": {
                   "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/workspace/someworkspace"
```

> **See Also:**
>
> *Oracle REST Data Services Installation, Configuration, and Development Guide* for more information on how to build a query.

32.2.2 Fetch Workspace-Level Statistics

This service returns usage statistics for a specific Application Express workspace.

HTTP Request Syntax Parameter

Table 32-3 HTTP Request Syntax

Parameter	Description
HTTP Method	`GET`

Table 32-3 (Cont.) HTTP Request Syntax

Parameter	Description
URL	`http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/workspace-name`
HTTP Request Headers	`"authorization": "Bearer: OAuth access token aquired with Authentication`

Returns

```
{
    "items": [
        {
            "log_day": "2016-09-15T00:00:00Z",
            "workspace_id": 1809919633676005,
            "application_id": 106,
            "application_name": "Sample Calendar",
            "application_link": {
                "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106"
            },
            "page_events": 94,
    :
```

Parameters

Filtering is possible for each of the JSON attributes by appending a query to the request URL as follows:

```
http://application-express-host.../latest/instance?q=query
```

Examples

Get All Statistics For Workspace "MYWORKSPACE"

The example displays the following output when you execute command line utility `curl`:

```
   $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/stats/
latest/workspace/myworkspace
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

> **Note:**
>
> The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
   select apex_web_service.make_rest_request(
           p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/workspace/myworkspace'
           p_http_method => 'GET',
           p_scheme =>      'OAUTH_CLIENT_CRED' )
   from dual;
```

The following is a JSON-Document response:

```
   {
     "items": [
        {
           "log_day": "2016-09-15T00:00:00Z",
           "workspace_id": 1809919633676005,
           "application_id": 106,
           "application_name": "Sample Calendar",
           "application_link": {
              "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106"
           },
           "page_events": 94,
   :
```

Get Workspace Statistics For "MYWORKSPACE" Since August 1st, 2016

The example displays the following output when you execute command line utility `curl`:

```
   $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/stats/
latest/workspace/myworkspace?q=%7B%22log_day%22:%7B%22$gt%22:%7B%22$date
%22:%222016-08-01T00:00:00Z%22%7D%7D%7D
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

Note:

The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
    select apex_web_service.make_rest_request(
           p_url =>      'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/workspace/myworkspace?q='||
                         utl_url.escape(' "log_day": "$gt":"$date":
"2016-08-01T00:00:00Z" '),
           p_http_method => 'GET' )
   from dual;
```

The following is a JSON-Document response:

```
  {
     "items": [
        {
           "log_day": "2016-09-15T00:00:00Z",
           "workspace_id": 1809919633676005,
           "application_id": 106,
           "application_name": "Sample Calendar",
           "application_link": {
```

```
            "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106"
          },
          "page_events": 94,
   :
```

See Also:

Oracle REST Data Services Installation, Configuration, and Development Guide for more information on how to build a query.

32.2.3 Application-Level Statistics

This service returns usage statistics for a specific application.

HTTP Request Syntax Parameter

Table 32-4 HTTP Request Syntax

Parameter	Description
HTTP Method	`GET`
URL	`http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/application-id`
HTTP Request Headers	`"authorization": "Bearer: OAuth access token aquired with Authentication"`

Returns

```
{
    "items": [
        {
            "log_day": "2016-09-15T00:00:00Z",
            "workspace_id": 1809919633676005,
            "application_id": 106,
            "application_name": "Sample Calendar",
            "application_link": {
                "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106"
            },
            "page_events": 94,
    :
```

Parameters

Filtering is possible for each of the JSON attributes by appending a query to the request URL as follows:

`http://application-express-host.../stats/latest/application/application-id?q=query`

Examples

Get All Statistics For Application "106"

The example displays the following output when you execute command line utility `curl`:

```
   $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/stats/
latest/application/106
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

> **Note:**
>
> The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
 select apex_web_service.make_rest_request(
           p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106'
           p_http_method => 'GET',
           p_scheme =>      'OAUTH_CLIENT_CRED' )
   from dual;
```

The following is a JSON-Document response:

```
  {
     "items": [
        {
           "log_day": "2016-09-15T00:00:00Z",
           "workspace_id": 1809919633676005,
           "application_id": 106,
           "application_name": "Sample Calendar",
           "application_link": {
              "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106"
           },
           "page_events": 94,
   :
```

Get Statistics For Application "106" Since August 1st, 2016

The example displays the following output when you execute command line utility `curl`:

```
    $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/stats/
latest/application/106?q=%7B%22log_day%22:%7B%22$gt%22:%7B%22$date
%22:%222016-08-01T00:00:00Z%22%7D%7D%7D
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

Note:

The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
    select apex_web_service.make_rest_request(
           p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106?q=' ||
                            utl_url.escape(' "log_day": "$gt": "$date":
"2016-08-01T00:00:00Z" '),
           p_http_method => 'GET' )
   from dual;
```

The following is a JSON-Document response:

```
  {
     "items": [
        {
           "log_day": "2016-09-15T00:00:00Z",
           "workspace_id": 1809919633676005,
           "application_id": 106,
           "application_name": "Sample Calendar",
           "application_link": {
              "$ref": "http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/application/106"
           },
           "page_events": 94,
   :
```

See Also:

Oracle REST Data Services Installation, Configuration, and Development Guide for more information on how to a build query.

32.2.4 Instance Overview

This service returns aggregated overview data for an Application Express instance.

HTTP Request Syntax Parameter

Table 32-5 HTTP Request Syntax

Parameter	Description
HTTP Method	`GET`
URL	`http://application-express-host:port/ords/apex_instance_admin_user/info/latest/instance/number-of-days`
HTTP Request Headers	`"authorization": "Bearer: OAuth access token aquired with Authentication`

Returns

```
{
    "items": [
        {
            "workspaces_total": 1074,
            "apps_total": 2827,
            "schemas_total": 1065,
            "reporting_timeframe_since": "2016-07-16T13:30:40Z",
            "reporting_timeframe_to": "2016-10-14T13:30:40Z",
            "active_apps_timeframe": 2827,
            "active_developers_timeframe": 731,
            "workspaces_timeframe": 1074
        }
    ],
    "first": {
        "$ref": "https://apexea.oracle.com/pls/apex/apex_instance_admin_user/info/
latest/instance/90"
    }
}
```

Parameters

`number-of-days`: Return the aggregated values from today to the given number of days into the past.

Example

The example displays the following output when you execute command line utility `curl`:

```
   $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/info/
latest/instance/90
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

> **Note:**
>
> The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
   select apex_web_service.make_rest_request(
           p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/info/latest/instance/90',
           p_http_method => 'GET' )
   from dual;
```

The following is a JSON-Document response:

```
{
    "items": [
        {
            "workspaces_total": 1074,
            "apps_total": 2827,
            "schemas_total": 1065,
            "reporting_timeframe_since": "2016-07-16T13:30:40Z",
```

```
            "reporting_timeframe_to": "2016-10-14T13:30:40Z",
            "active_apps_timeframe": 2827,
            "active_developers_timeframe": 731,
            "workspaces_timeframe": 1074
        }
    ],
    "first": {
        "$ref": "https://apexea.oracle.com/pls/apex/apex_instance_admin_user/info/
latest/instance/90"
    }
}
```

32.2.5 REST Service Version Information

This service returns version information for the REST Administrative Interface.

HTTP Request Syntax Parameter

Table 32-6 HTTP Request Syntax

Parameter	Description
HTTP Method	`GET`
URL	`http://`*`application-express-host`*`:`*`port`*`/ords/apex_instance_admin_user/info/latest/`
HTTP Request Headers	`"authorization": "Bearer: `*`OAuth access token aquired with Authentication`*

Returns

```
{
    "items": [
        {
            "version": "5.1.0"
        }
    ]
}
```

Example

The example displays the following output when you execute command line utility `curl`:

```
 $ curl -H"Authorization: Bearer LfXJilIBdzj5JPRn4xb5QQ..
          -i http://localhost:8081/ords/restauth/emp/list/
          http://application-express-host:port/ords/apex_instance_admin_user/info/
rest-service-version
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

Note:

The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.

```
      select apex_web_service.make_rest_request(
           p_url =>         'http://application-express-host:port/ords/
apex_instance_admin_user/info/latest/rest-service-version',
           p_http_method => 'GET' )
   from dual;
```

The following is a JSON-Document response:

```
{
    "items": [
        {
            "version": "5.1.0"
        }
    ]
}
```

Index

A

C

E

F

G

I

J

L

N

P

R

S

U

V

W