Oracle® Application Express

API Reference

Release 3.1.2

E12855-01

August 2008

ORACLE®

Oracle Application Express API Reference, Release 3.1.2

E12855-01

Primary Author: Drue Baker

Contributors: Marco Adelfio, Drue Baker, Carl Backstrom, Christina Cho, Steve Fogel, Michael Hichwa, Christopher Jones, Joel Kallman, Sharon Kennedy, Syme Kutz, Sergio Leunissen, Anne Romano, Kris Rice, Marc Sewtz, Scott Spadafore, Scott Spendolini, Jason Straub, Simon Watt, and Terri Winters

Contents

Index

Preface

Oracle Application Express API Reference describes the Application Programming Interfaces, referred to as APIs, available when programming in the Oracle Application Express environment.

This preface contains these topics:

Topic Overview

This document contains the following chapters:

Title	Description
APEX_UTIL	Use the `APEX_UTIL` package to get and set session state, get files, check authorizations for users, reset different states for users, and also to get and set preferences for users.
APEX_MAIL	Use the `APEX_MAIL` package to send an email from an Oracle Application Express application.
APEX_ITEM	Use the `APEX_ITEM` package to create form elements dynamically based on a SQL query instead of creating individual items page by page.
APEX_APPLICATION	Use the `APEX_APPLICATION` package to take advantage of a number of global variables.
APEX_CUSTOM_AUTH	Use the `APEX_CUSTOM_AUTH` package to perform various operations related to authentication and session management.
APEX_LDAP	Use `APEX_LDAP` to perform various operations related to Lightweight Directory Access Protocol (LDAP) authentication.
APEX_INSTANCE_ADMIN	Use the `APEX_INSTANCE_ADMIN` package to get and set email settings, wallet settings, report printing settings and to manage schema to workspace mappings.

Title	Description
APEX_UI_DEFAULT_UPDATE	You can use the `APEX_UI_DEFAULT_UPDATE` 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.
JavaScript APIs	Use these JavaScript functions and objects to provide client-side functionality, such as showing and hiding page elements, or making XML HTTP Asynchronous JavaScript and XML (AJAX) requests.

Note: In release 2.2, Oracle Application Express APIs were renamed using the prefix `APEX_`. Note that API's using the previous prefix `HTMLDB_` are still supported to provide backward compatibility. As a best practice, however, use the new API names for new applications unless you plan to run them in an earlier version of Oracle Application Express.

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 as well as an understanding of the operating system environment under which you are running Oracle Application Express.

See Also: *Oracle 2 Day + Application Express Developer's Guide*

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents

For more information, see these Oracle resources:

- *Oracle Application Express Release Notes*
- *Oracle Application Express Installation Guide*
- *Oracle 2 Day + Application Express Developer's Guide*
- *Oracle Application Express Advanced Tutorials*
- *Oracle Application Express Administration Guide*
- *Oracle Application Express Migration Guide*
- *Oracle Application Express SQL Workshop and Utilities 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*

For information about Oracle error messages, see *Oracle Database Error Messages*. Oracle error message documentation is available only in HTML. If you have access to the Oracle Database Documentation Library, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

http://www.oracle.com/technology/documentation/

Conventions

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.

1

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, and also to get and set preferences for users.

Topics in this section include:

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 1–1 describes the parameters available in the CHANGE_CURRENT_USER_PW procedure.

Table 1–1 CHANGE_CURRENT_USER_PW Parameters

Parameter	Description
p_new_password	The new password value in clear text

Example

```
BEGIN
APEX_UTIL.CHANGE_CURRENT_USER_PW ('secret99');
END;
```

CACHE_GET_DATE_OF_PAGE_CACHE Procedure

This procedure 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 1–2 describes the parameters available in the CACHE_GET_DATE_OF_PAGE_CACHE procedure.

Table 1–2 CACHE_GET_DATE_OF_PAGE_CACHE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The page number (ID).

CACHE_GET_DATE_OF_REGION_CACHE Procedure

This procedure 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 1–3 describes the parameters available in the `CACHE_GET_DATE_OF_REGION_CACHE` procedure.

Table 1–3 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

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 1–4 describes the parameters available in the CACHE_PURGE_BY_APPLICATION procedure.

Table 1–4 CACHE_PURGE_BY_APPLICATION Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

CACHE_PURGE_BY_PAGE Procedure

This procedure purges all cached pages and regions for a given application and page.

Syntax

```
APEX_UTIL.CACHE_PURGE_BY_PAGE (
    p_application  IN    NUMBER,
    p_page         IN    NUMBER,
    p_user_name    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–5 describes the parameters available in the CACHE_PURGE_BY_PAGE procedure.

Table 1–5 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.

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 will no longer be used, thus removing those unusable pages or regions from the cache.

Syntax

```
APEX_UTIL.CACHE_PURGE_STALE (
    p_application  IN    NUMBER,
```

Parameters

Table 1–6 describes the parameters available in the CACHE_PURGE_STALE procedure.

Table 1–6 CACHE_PURGE_STALE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

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. 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 1–7 describes the parameters available in the `CHANGE_PASSWORD_ON_FIRST_USE` function.

Table 1–7 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 Application Express user account (workspace administrator, developer, or end user) in the current workspace must changed by the user the first time it is used.

```
BEGIN
  FOR c1 IN (SELECT user_name FROM wwv_flow_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" on page 1-66

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 1–8 describes the parameters available in the CLEAR_APP_CACHE procedure.

Table 1–8 CLEAR_APP_CACHE Parameters

Parameter	Description
p_app_id	The ID of the application for which session state will be cleared for current session

Example

```
BEGIN
        APEX_UTIL.CLEAR_APP_CACHE('100');
END;
```

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 IN NUMBER DEFAULT NULL);
```

Parameters

Table 1–9 describes the parameters available in the CLEAR_PAGE_CACHE procedure.

Table 1–9 CLEAR_PAGE_CACHE Parameters

Parameter	Description
p_page	The ID of the page in the current application for which session state will be cleared for current session

Example

```
BEGIN
APEX_UTIL.CLEAR_PAGE_CACHE('10');
END;
```

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

```
BEGIN
       APEX_UTIL.CLEAR_USER_CACHE;
END;
```

COUNT_CLICK Procedure

This procedure counts clicks from an application built in Application 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 1–10 describes the parameters available in the `COUNT_CLICK` procedure.

Table 1–10 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)
`p_workspace`	The workspace associated with the application (optional)

Example

```
BEGIN
htp.p('<a
href=APEX_UTIL.COUNT_CLICK?p_url=http://yahoo.com&p_cat=yahoo&p_workspace=NNN>
Click</a>');
end;
```

Where `NNN` equals your workspace ID.

> **See Also:** "Purging the External Click Count Log" in *Oracle Application Express Administration Guide*

COUNT_STALE_REGIONS Function

Counts the number of expired regions.

Syntax

```
APEX_UTIL.COUNT_STALE_REGIONS (
     p_application IN NUMBER,
RETURN NUMBER;
```

Parameters

Table 1–11 describes the parameters available in COUNT_STALE_REGIONS.

Table 1–11 COUNT_STALE_REGIONS Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

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                      NUMBER                 IN     DEFAULT NULL,
    p_user_name                    VARCHAR2               IN,
    p_first_name                   VARCHAR2               IN     DEFAULT NULL,
    p_last_name                    VARCHAR2               IN     DEFAULT NULL,
    p_description                  VARCHAR2               IN     DEFAULT NULL,
    p_email_address                VARCHAR2               IN     DEFAULT NULL,
    p_web_password                 VARCHAR2               IN,
    p_web_password_format          VARCHAR2               IN     DEFAULT NULL,
    p_group_ids                    VARCHAR2               IN     DEFAULT NULL,
    p_attribute_01                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_02                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_03                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_04                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_05                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_06                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_07                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_08                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_09                 VARCHAR2               IN     DEFAULT NULL,
    p_attribute_10                 VARCHAR2               IN     DEFAULT NULL);
```

Parameters

Table 1–12 describes the parameters available in the `CREATE_USER` procedure.

Table 1–12 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_address`	Clear text password
`p_group_ID`	Colon separated list of numeric group IDs
`p_attribute_01`	Arbitrary text accessible with an API
`...`	
`p_attribute_10`	

Example

```
BEGIN
APEX_UTIL.CREATE_USER(
    P_USER_NAME    => 'NEWUSER1',
```

```
    P_WEB_PASSWORD => 'secret99');
END;
```

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                    NUMBER                  IN,
    p_group_name            VARCHAR2                IN,
    p_security_group_id     NUMBER                  IN,
    p_group_desc            VARCHAR2                IN);
```

Parameter

Table 1–13 describes the parameters available in the `CREATE_USER_GROUP` procedure.

Table 1–13 CREATE_USER_GROUP Parameters

Parameter	Description
`p_id`	Primary key of group
`p_group_name`	Arbitrary name
`p_security_group_id`	Workspace ID
`p_group_desc`	Descriptive text

Example

```
BEGIN
APEX_UTIL.CREATE_USER_GROUP (
    p_id               => 0 - trigger will assign PK,
    p_group_name       => 'Managers',
    p_security_group_id => null, -- defaults to current workspace ID
    p_group_desc       => 'text');
END;
```

CURRENT_USER_IN_GROUP Function

This function returns a Boolean result based on whether or not 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 1–14 describes the parameters available in the `CURRENT_USER_IN_GROUP` function.

Table 1–14 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

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := APEX_UTIL.CURRENT_USER_IN_GROUP(p_group_name=>'Managers');
END;
```

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

```
EDIT_USER (
    p_user_id                    NUMBER                 IN,
    p_user_name                  VARCHAR2               IN,
    p_first_name                 VARCHAR2               IN     DEFAULT NULL,
    p_last_name                  VARCHAR2               IN     DEFAULT NULL,
    p_web_password               VARCHAR2               IN     DEFAULT NULL,
    p_new_password               VARCHAR2               IN     DEFAULT NULL,
    p_email_address              VARCHAR2               IN     DEFAULT NULL,
    p_start_date                 VARCHAR2               IN     DEFAULT NULL,
    p_end_date                   VARCHAR2               IN     DEFAULT NULL,
    p_employee_id                VARCHAR2               IN     DEFAULT NULL,
    p_allow_access_to_schemas    VARCHAR2               IN     DEFAULT NULL,
    p_person_type                VARCHAR2               IN     DEFAULT NULL,
    p_default_schema             VARCHAR2               IN     DEFAULT NULL,
    p_group_idS                  VARCHAR2               IN     DEFAULT NULL,
    p_developer_roles            VARCHAR2               IN     DEFAULT NULL,
    p_description                VARCHAR2               IN     DEFAULT NULL,
    p_account_expiry             DATE                   IN     DEFAULT NULL,
    p_account_locked             VARCHAR2               IN     DEFAULT 'N',
    p_failed_access_attempts     NUMBER                 IN     DEFAULT 0,
    p_change_password_on_first_use VARCHAR2             IN     DEFAULT 'Y',
    p_first_password_use_occurred VARCHAR2              IN     DEFAULT 'N');
```

Parameters

Table 1–15 describes the parameters available in the EDIT_USER procedure.

Table 1–15 EDIT_USER Parameters

Parameter	Description
p_user_id	Numeric primary key of the user account
p_user_name	Alphanumeric name used for login
p_first_name	Informational
p_last_name	Informational
p_web_password	Clear text password
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

Table 1–15 (Cont.) EDIT_USER Parameters

Parameter	Description
p_developer_privs	Colon-separated list of developer privileges (only ADMIN: has meaning to Oracle Application Express)
p_description	Informational
p_account_expiry	Date password was last updated.
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.
p_change_password_on_first_use	'Y' or 'N' to indicate whether password must be changed on first use.
p_first_password_use_occurred	'Y' or 'N' to indicate whether login has occurred since password change

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 1–16 describes the parameters available in the END_USER_ACCOUNT_DAYS_LEFT function.

Table 1–16 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 will expire.

```
DECLARE
  l_days_left NUMBER;
BEGIN
  FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
    l_days_left := APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT(p_user_name => c1.user_
name) THEN
    htp.p('End User Account:'||c1.user_name||' will expire in '||l_days_left||'
days.');
  END LOOP;
END;
```

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 1–18 describes the parameters available in the EXPIRE_END_USER_ACCOUNT procedure.

Table 1–17 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 with respect to its use by end users to authenticate to developed applications, but it may also expire the account with respect to 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 wwv_flow_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;
```

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 1–18 describes the parameters available in the EXPIRE_WORKSPACE_ACCOUNT procedure.

Table 1–18 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 with respect to its use by developers or administrators to log in to a workspace, but it may also expire the account with respect to its use by end users to authenticate to developed applications.

```
BEGIN
  FOR c1 IN (SELECT user_name FROM wwv_flow_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;
```

EXPORT_USERS Procedure

When called from an 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 1–19 describes the parameters available in the EXPORT_USERS procedure.

Table 1–19 EXPORT_USERS Parameters

Parameter	Description
p_export_format	Indicates how rows in the export file will be 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

```
BEGIN
  APEX_UTIL.EXPORT_USERS;
END;
```

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 1–20 describes the parameters available in the FETCH_APP_ITEM function.

Table 1–20 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

```
DECLARE VAL VARCHAR2(30);
BEGIN
VAL := APEX_UTIL.FETCH_APP_ITEM (p_item=>'F300_NAME',p_app=>300);
END;
```

FETCH_USER Procedure

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.

Fetch_user Procedure Signature 1:

Syntax

```
FETCH_USER (
    p_user_id                   NUMBER                  IN,
    p_workspace                 VARCHAR2                OUT,
    p_user_name                 VARCHAR2                OUT,
    p_first_name                VARCHAR2                OUT,
    p_last_name                 VARCHAR2                OUT,
    p_web_password              VARCHAR2                OUT,
    p_email_address             VARCHAR2                OUT,
    p_start_date                VARCHAR2                OUT,
    p_end_date                  VARCHAR2                OUT,
    p_employee_id               VARCHAR2                OUT,
    p_allow_access_to_schemas   VARCHAR2                OUT,
    p_person_type               VARCHAR2                OUT,
    p_default_schema            VARCHAR2                OUT,
    p_groups                    VARCHAR2                OUT,
    p_developer_role            VARCHAR2                OUT,
    p_description               VARCHAR2                OUT );
```

Parameters

Table 1–21 describes the parameters available in the FETCH_USER procedure.

Table 1–21 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
p_first_name	Informational
p_last_name	Informational
p_web_password	Obfuscated account password
p_email_address	Email address
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

Table 1–21 (Cont.) Fetch_User Parameters Signature 1

Parameter	Description
p_groups	List of groups of which user is a member
p_developer_role	Unused
p_description	Informational

Fetch_user Procedure Signature 2:

Syntax

```
FETCH_USER (
    p_user_id                      NUMBER                  IN,
    p_workspace                    VARCHAR2                OUT,
    p_user_name                    VARCHAR2                OUT,
    p_first_name                   VARCHAR2                OUT,
    p_last_name                    VARCHAR2                OUT,
    p_email_address                VARCHAR2                OUT,
    p_groups                       VARCHAR2                OUT,
    p_developer_role               VARCHAR2                OUT,
    p_description                  VARCHAR2                OUT );
```

Parameters

Table 1–22 describes the parameters available in the FETCH_USER procedure.

Table 1–22 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
p_first_name	Informational
p_last_name	Informational
p_web_password	Obfuscated account password
p_email_address	Email address
p_groups	List of groups of which user is a member
p_developer_role	Unused
p_description	Informational

FETCH_USER Procedure Signature 3

Syntax

```
FETCH_USER (
    p_user_id                      NUMBER                  IN,
    p_workspace                    VARCHAR2                OUT,
    p_user_name                    VARCHAR2                OUT,
    p_first_name                   VARCHAR2                OUT,
    p_last_name                    VARCHAR2                OUT,
    p_web_password                 VARCHAR2                OUT,
    p_email_address                VARCHAR2                OUT,
    p_start_date                   VARCHAR2                OUT,
    p_end_date                     VARCHAR2                OUT,
```

```
p_employee_id                  VARCHAR2                OUT,
p_allow_access_to_schemas      VARCHAR2                OUT,
p_person_type                  VARCHAR2                OUT,
p_default_schema               VARCHAR2                OUT,
p_groups                       VARCHAR2                OUT,
p_developer_role               VARCHAR2                OUT,
p_account_expiry               DATE                    OUT,
p_account_locked               VARCHAR2                OUT,
p_change_password_on_first_use VARCHAR2                OUT,
p_first_password_use_occurred  VARCHAR2                OUT);
```

Parameters

Table 1–23 describes the parameters available in the FETCH_USER procedure.

Table 1–23 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
p_first_name	Informational
p_last_name	Informational
p_web_password	Obfuscated account password
p_email_address	Email address
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
p_groups	List of groups of which user is a member
p_developer_role	Unused
p_description	Informational
p_account_expiry	Date account password was last reset
p_account_locked	Locked/Unlocked indicator Y or N
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

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 1–24 describes the parameters available in the `FIND_SECURITY_GROUP_ID` function.

Table 1–24 FIND_SECURITY_GROUP_ID Parameters

Parameter	Description
`p_workspace`	The name of the workspace

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := APEX_UTIL.FIND_SECURITY_GROUP_ID (p_workspace=>'DEMOS');
END;
```

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 1–25 describes the parameters available in the `FIND_WORKSPACE` function.

Table 1–25 FIND_WORKSPACE Parameters

Parameter	Description
`p_security_group_id`	The security group ID of a workspace

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := APEX_UTIL.FIND_WORKSPACE (p_security_group_id =>'20');
END;
```

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 1–26 describes the parameters available in the GET_ACCOUNT_LOCKED_STATUS function.

Table 1–26 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 wwv_flow_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;
```

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.

Syntax

```
APEX_UTIL.GET_ATTRIBUTE(
    p_username                IN VARCHAR2,
    p_attribute_number        IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 1–27 describes the parameters available in the `GET_ATTRIBUTE` function.

Table 1–27 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

```
DECLARE VAL VARCHAR2(30);
BEGIN
  VAL := APEX_UTIL.GET_ATTRIBUTE (
                          p_username => 'FRANK',
                          p_attribute_number => 1);
END;
```

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);
```

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.

> **See Also:** "About BLOB Support in Forms and Reports" in *Oracle Application Express Application Builder User's Guide*

Syntax

```
FUNCTION 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 1–28 describes the parameters available in GET_BLOB_FILE_SRC function.

Table 1–28 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.

> **Tip:** See "Running a Demonstration Application" in *Oracle Application Express Application Builder User's Guide*.

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

```
DECLARE VAL NUMBER;
BEGIN
  VAL := APEX_UTIL.GET_CURRENT_USER_ID;
END;
```

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

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL. GET_DEFAULT_SCHEMA;
END;
```

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 1–29 describes the parameters available in GET_EMAIL function.

Table 1–29 GET_EMAIL Parameters

Parameter	Description
p_username	The user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_EMAIL(p_username => 'FRANK');
END;
```

GET_FILE Procedure

This procedure downloads files from the Oracle Application Express file repository.

Syntax

```
APEX_UTIL.GET_FILE (
    p_file_id    IN   VARCHAR2,
    p_mime_type  IN   VARCHAR2 DEFAULT NULL,
    p_inline     IN   VARCHAR2 DEFAULT 'NO');
```

Parameters

Table 1–30 describes the parameters available in GET_FILE procedure.

Table 1–30 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_mime_type => 'text/xml',`
	`  p_inline => 'YES');`
	`END;`

p_mime_type	Mime type of the file to download
p_inline	Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment

Example

```
BEGIN
        APEX_UTIL.GET_FILE(
              p_file_id   => '8675309',
              p_mime_type => 'text/xml',
              p_inline    => 'YES');
END;
```

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_fname    IN   VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 1–31 describes the parameters available in GET_FILE_ID function.

Table 1–31 GET_FILE_ID Parameters

Parameter	Description
p_fname	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

```
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_fname => l_name);
END;
```

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 1–32 describes the parameters available in GET_FIRST_NAME function.

Table 1–32 GET_FIRST_NAME Parameters

Parameter	Description
p_username	Identifies the user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_FIRST_NAME(p_username => 'FRANK');
END;
```

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 1–33 describes the parameters available in GET_GROUPS_USER_BELONGS_TO function.

Table 1–33 GET_GROUPS_USER_BELONGS_TO Parameters

Parameter	Description
p_username	Identifies the user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'FRANK');
END;
```

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)
RETURN VARCHAR2;
```

Parameters

Table 1–34 describes the parameters available in GET_GROUP_ID function.

Table 1–34 GET_GROUP_ID Parameters

Parameter	Description
p_group_name	Identifies the user name in the account

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := APEX_UTIL.GET_GROUP_ID(p_group_name => 'Managers');
END;
```

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)
RETURN NUMBER;
```

Parameters

Table 1–35 describes the parameters available in GET_GROUP_NAME function.

Table 1–35 GET_GROUP_NAME Parameters

Parameter	Description
p_group_id	Identifies a numeric ID of a group in the workspace

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;
```

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 1–36 describes the parameters available in GET_LAST_NAME function.

Table 1–36 GET_LAST_NAME Parameters

Parameter	Description
p_username	The user name in the user account record

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_LAST_NAME(p_username => 'FRANK');
END;
```

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)
RETURN VARCHAR2;
```

Parameters

Table 1–37 describes the parameters available in `GET_USERNAME` function.

Table 1–37 GET_USERNAME Parameters

Parameter	Description
`p_userid`	Identifies the numeric ID of a user account in the workspace

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_USERNAME(p_userid => 228922003);
END;
```

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 1–38 describes the parameters available in GET_NUMERIC_SESSION_STATE function.

Table 1–38 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

```
DECLARE
      l_item_value    Number;
BEGIN
      l_item_value := APEX_UTIL.GET_NUMERIC_SESSION_STATE('my_item');
END;
```

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 1–39 describes the parameters available in the GET_PREFERENCE function.

Table 1–39 GET_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference to retrieve the value
p_value	Value of the preference
p_user	User for whom the preference is being retrieved

Example

```
DECLARE
      l_default_view    VARCHAR2(255);
BEGIN
      l_default_view := APEX_UTIL.GET_PREFERENCE(
                  p_preference => 'default_view',
                  p_user       => :APP_USER);
END;
```

GET_PRINT_DOCUMENT Function

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 1–40 describes the parameters available in the `GET_PRINT_DOCUMENT` function.

Table 1–40 GET_PRINT_DOCUMENT 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 will be derived from preferences.

GET_PRINT_DOCUMENT Function

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_layout_name  IN VARCHAR2,
    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 1–41 describes the parameters available in the `GET_PRINT_DOCUMENT` function.

Table 1–41 GET_PRINT_DOCUMENT Parameters

Parameter	Description
`p_application_id`	Defines the application ID of the report query
`p_report_layout_name`	Name of the report layout (stored under application's Shared Components)
`p_report_query_name`	Name of the report query (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 will be derived from preferences.

GET_PRINT_DOCUMENT Function

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 1–42 describes the parameters available in the `GET_PRINT_DOCUMENT` function.

Table 1–42 GET_PRINT_DOCUMENT 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 will be derived from preferences.

GET_PRINT_DOCUMENT Function

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 1–43 describes the parameters available in the GET_PRINT_DOCUMENT function.

Table 1–43 GET_PRINT_DOCUMENT 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 will be derived from preferences

DOWNLOAD_PRINT_DOCUMENT Procedure

This procedure initiates the download of a print document using XML based report data 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 1–44 describes the parameters available in the `DOWNLOAD_PRINT_DOCUMENT` function.

Table 1–44 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
`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 will be derived from preferences.

DOWNLOAD_PRINT_DOCUMENT Procedure

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 1–45 describes the parameters available in the `DOWNLOAD_PRINT_DOCUMENT` function.

Table 1–45 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`	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 will be derived from preferences.

DOWNLOAD_PRINT_DOCUMENT Procedure

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 1–46 describes the parameters available in the DOWNLOAD_PRINT_DOCUMENT function.

Table 1–46 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 will be derived from preferences.

DOWNLOAD_PRINT_DOCUMENT Procedure

This procedure initiates the download of a print document using XML based report data 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_query_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 1–46 describes the parameters available in the `DOWNLOAD_PRINT_DOCUMENT` function.

Table 1–47 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 will be derived from preferences.

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 1–48 describes the parameters available in `GET_SESSION_STATE` function.

Table 1–48 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

```
DECLARE
      l_item_value  VARCHAR2(255);
BEGIN
      l_item_value := APEX_UTIL.GET_SESSION_STATE('my_item');
END;
```

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)
RETURN VARCHAR2;
```

Parameters

Table 1–49 describes the parameters available in `GET_USER_ID` function.

Table 1–49 GET_USER_ID Parameters

Parameter	Description
`p_username`	Identifies the name of a user in the workspace

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := APEX_UTIL.GET_USER_ID(p_username => 'Managers');
END;
```

GET_USER_ROLES Function

This function returns the DEVELOPER_ROLE field stored in the named user account record.

Syntax

```
APEX_UTIL.GET_USER_ROLES(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 1–50 describes the parameters available in GET_USER_ROLES function.

Table 1–50 GET_USER_ROLES Parameters

Parameter	Description
p_username	Identifies a user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := APEX_UTIL.GET_USER_ROLES(p_username=>'FRANK');
END;
```

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 1–51 describes the parameters available in the `IS_LOGIN_PASSWORD_VALID` function.

Table 1–51 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

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := APEX_UTIL. IS_LOGIN_PASSWORD_VALID (
             p_username=>'FRANK'
             p_password=>'tiger');
END;
```

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 1–52 describes the parameters available in `IS_USERNAME_UNIQUE` function.

Table 1–52 IS_USERNAME_UNIQUE Parameters

Parameter	Description
`p_username`	Identifies the user name to be tested

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := APEX_UTIL.IS_USERNAME_UNIQUE(
             p_username=>'FRANK');
END;
```

KEYVAL_NUM Function

This function gets the value of the package variable (`wwv_flow_utilities.g_val_num`) set by `APEX_UTIL.SAVEKEY_NUM`.

Syntax

```
APEX_UTIL.KEYVAL_NUM;
```

Parameters

Table 1–53 describes the parameters available in `KEYVAL_NUM` function.

Table 1–53 KEYVAL_NUM Parameters

Parameter	Description
`p_val`	The numeric value previously saved

Example

```
DECLARE
VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.KEYVAL_NUM;
END;
```

See Also: "SAVEKEY_NUM Function" on page 1-79

KEYVAL_VC2 Function

This function gets the value of the package variable (`wwv_flow_utilities.g_val_vc2`) set by `APEX_UTIL.SAVEKEY_VC2`.

Syntax

```
APEX_UTIL.KEYVAL_VC2;
```

Parameters

`p_val` is the VARCHAR2 value previously saved.

Example

```
DECLARE
VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.KEYVAL_VC2;

END;
```

See Also: "SAVEKEY_VC2 Function" on page 1-80

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 1–54 describes the parameters available in the LOCK_ACCOUNT procedure.

Table 1–54 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 wwv_flow_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;
```

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 1–55 describes the parameters available in the PASSWORD_FIRST_USE_OCCURRED procedure.

Table 1–55 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 wwv_flow_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" on page 1-10

PREPARE_URL Function

Given a ready-to-render f?p relative URL, this function adds a Session State Protection checksum argument (&cs=) if one is required.

> **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 in order 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)
RETURN VARCHAR2;
```

Parameters

Table 1–56 describes the parameters available in the PREPARE_URL function.

Table 1–56 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

Example

```
DECLARE
l_url varchar2(2000);
l_session number := v('APP_SESSION');
BEGIN
l_url :=
APEX_UTIL.PREPARE_URL('f?p=100:1:'||l_session||'::NO::P1_ITEM:xyz');
END;
```

PUBLIC_CHECK_AUTHORIZATION Function

Given the name of a security 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 1–57 describes the parameters available in `PUBLIC_CHECK_AUTHORIZATION` function.

Table 1–57 PUBLIC_CHECK_AUTHORIZATION Parameters

Parameter	Description
`p_security_name`	The name of the security scheme that determines if the user passes the security check

Example

```
DECLARE
      l_check_security  BOOLEAN;
BEGIN
      l_check_security := APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION('my_auth_scheme');
END;
```

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 1–58 describes the parameters available in PURGE_REGIONS_BY_APP.

Table 1–58 PURGE_REGIONS_BY_APP Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

```
APEX_UTILITIES.PURGE_REGIONS_BY_APP(p_application=>123);
```

PURGE_REGIONS_BY_ID Procedure

Deletes all cached values for a region.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_ID (
     p_application IN NUMBER,
     p_region_id   IN NUMBER);
```

Parameters

Table 1–59 describes the parameters available in PURGE_REGIONS_BY_ID.

Table 1–59 PURGE_REGIONS_BY_ID Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_region_id	The identification number of the region for which cached values are deleted.

PURGE_REGIONS_BY_NAME Procedure

Deletes all cached regions identified by the application name and page number.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_NAME (
     p_application  IN NUMBER,
     p_page      IN NUMBER,
     p_region_name  IN VARCHAR2);
```

Parameters

Table 1–60 describes the parameters available in PURGE_REGIONS_BY_NAME.

Table 1–60 PURGE_REGIONS_BY_NAME Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The number of the page containing the region to be deleted.
p_region_name	The region to be deleted.

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 1–61 describes the parameters available in `PURGE_REGIONS_BY_PAGE`.

Table 1–61 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.

PURGE_STALE_REGIONS Procedure

Deletes all cached regions that have expired or are no longer useful.

Syntax

```
APEX_UTIL.PURGE_STALE_REGIONS (
     p_application IN NUMBER,
```

Parameters

Table 1–62 describes the parameters available in PURGE_STALE_REGIONS.

Table 1–62 PURGE_STALE_REGIONS Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

REMOVE_PREFERENCE Procedure

This function 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 1–63 describes the parameters available in the REMOVE_PREFERENCE procedure.

Table 1–63 REMOVE_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference to remove
p_user	User for whom the preference is defined

Example

```
BEGIN
       APEX_UTIL.REMOVE_PREFERENCE(
                    p_preference => 'default_view',
                    p_user       => :APP_USER);
END;
```

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 1–64 describes the parameters available in `REMOVE_SORT_PREFERENCES` function.

Table 1–64 REMOVE_SORT_PREFERENCES Parameters

Parameter	Description
`p_user`	Identifies the user for whom sorting preferences will be removed

Example

```
BEGIN
      APEX_UTIL.REMOVE_SORT_PREFERENCES(:APP_USER);
END;
```

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 1–65 describes the parameters available in the REMOVE_USER procedure.

Table 1–65 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

```
BEGIN
APEX_UTIL.REMOVE_USER(p_user_id=>'99997');
END;

BEGIN
APEX_UTIL.REMOVE_USER(p_user_name => 'FRANK');
END;
```

RESET_AUTHORIZATIONS 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

```
BEGIN
APEX_UTIL.RESET_AUTHORIZATIONS;
END;
```

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 1–66 describes the parameters available in the `RESET_PW` procedure.

Table 1–66 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

```
BEGIN
APEX_UTIL.RESET_PW(
    p_user => 'FRANK',
    p_msg => 'Contact help desk at 555-1212 with questions');
END;
```

SAVEKEY_NUM Function

This function sets a package variable (`wwv_flow_utilities.g_val_num`) so that it can be retrieved using the function `KEYVAL_NUM`.

Syntax

```
APEX_UTIL.SAVEKEY_NUM(
    p_val IN NUMBER);
```

Parameters

Table 1–67 describes the parameters available in the `SAVEKEY_NUM` procedure.

Table 1–67 SAVEKEY_NUM Parameters

Parameter	Description
`p_val`	The numeric value to be saved

Example

```
DECLARE
VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.SAVEKEY_NUM(
        p_val => 10);
END;
```

See Also: "KEYVAL_NUM Function" on page 1-63

SAVEKEY_VC2 Function

This function sets a package variable (`wwv_flow_utilities.g_val_vc2`) so that it can be retrieved using the function `KEYVAL_VC2`.

Syntax

```
APEX_UTIL.SAVEKEY_VC2
    p_val IN VARCHAR2);
```

Parameters

Table 1–68 describes the parameters available in the `SAVEKEY_VC2` function.

Table 1–68 SAVEKEY_VC2 Parameters

Parameter	Description
`p_val`	The is the VARCHAR2 value to be saved

Example

```
DECLARE
VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.SAVEKEY_VC2(
        p_val => 'XXX');
END;
```

See Also: "KEYVAL_VC2 Function" on page 1-64

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 1–69 describes the parameters available in the `SET_ATTRIBUTE` procedure.

Table 1–69 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

```
DECLARE VAL VARCHAR2(30);
BEGIN
    APEX_UTIL.SET_ATTRIBUTE (
        p_userid => apex_util.get_user_id(p_username => 'FRANK'),
        p_attribute_number => 1,
        p_attribute_value => 'foo');
END;
```

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.

> **See Also:** "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*

Syntax

```
APEX_UTIL.SET_AUTHENTICATION_RESULT(
    p_code IN NUMBER
    );
```

Parameters

Table 1–18 describes the parameters available in the `SET_AUTHENTICATION_RESULT` procedure.

Table 1–70 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;
```

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.

> **See Also:** "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*

Syntax

```
APEX_UTIL.SET_CUSTOM_AUTH_STATUS(
    p_status  IN VARCHAR2
    );
```

Parameters

Table 1–71 describes the parameters available in the SET_CUSTOM_AUTH_STATUS procedure.

Table 1–71 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;
```

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 1–72 describes the parameters available in the SET_EMAIL procedure.

Table 1–72 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

```
BEGIN
APEX_UTIL.SET_EMAIL(
    p_userid  => '888883232',
    P_email   => 'frank.scott@oracle.com');
END;
```

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 1–73 describes the parameters available in the SET_FIRST_NAME procedure.

Table 1–73 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

```
BEGIN
APEX_UTIL.SET_FIRST_NAME(
    p_userid       => '888883232',
    P_first_name   => 'FRANK');
END;
```

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 1–74 describes the parameters available in the SET_LAST_NAME procedure.

Table 1–74 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

```
BEGIN
APEX_UTIL.SET_LAST_NAME(
    p_userid       => '888883232',
    p_last_name    => 'SMITH');
END;
```

SET_PREFERENCE Procedure

This procedure sets a preference that will persist 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 1–75 describes the parameters available in the `SET_PREFERENCE` procedure.

Table 1–75 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

```
BEGIN
       APEX_UTIL.SET_PREFERENCE(
             p_preference => 'default_view',
             p_value      => 'WEEKLY',
             p_user       => :APP_USER);
END;
```

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);
```

Parameters

Table 1–76 describes the parameters available in the `SET_SESSION_STATE` procedure.

Table 1–76 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

Example

```
BEGIN
APEX_UTIL.SET_SESSION_STATE('my_item','myvalue');
END;
```

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 1–77 describes the parameters available in the SET_USERNAME procedure.

Table 1–77 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

```
BEGIN
APEX_UTIL.SET_USERNAME(
    p_userid       => '888883232',
    P_username   => 'USER-XRAY');
END;
```

STRONG_PASSWORD_CHECK Procedure

This procedure returns `Boolean OUT` values based on whether or not 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)
    ;
```

Parameters

Table 1–78 describes the parameters available in the `STRONG_PASSWORD_CHECK` procedure.

Table 1–78 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
`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

Table 1–78 (Cont.) STRONG_PASSWORD_CHECK Parameters

Parameter	Description
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

Example

```
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;
BEGIN
    l_username := 'SOMEBODY';
    l_password := 'foo';
    l_old_password := 'foo';
    l_workspace_name := 'XYX_WS';

    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)

    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_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;

END;
```

STRONG_PASSWORD_VALIDATION Function

This function returns formatted HTML in a VARCHAR2 result based on whether or not 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 1–79 describes the parameters available in the `STRONG_PASSWORD_VALIDATION` function.

Table 1–79 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

```
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,

    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_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;

END;
```

STRING_TO_TABLE Function

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 1–80 describes the parameters available in the STRING_TO_TABLE function.

Table 1–80 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

```
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;
```

TABLE_TO_STRING Function

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 1–81 describes the parameters available in the TABLE_TO_STRING function.

Table 1–81 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

```
DECLARE
       l_string     VARCHAR2(255);
       l_vc_arr2    APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
       l_vc_arr2 := APEX_UTIL.STRING_TO_TABLE('One:Two:Three');

       l_string := APEX_UTIL.TABLE_TO_STRING(l_vc_arr2);
END;
```

UNEXPIRE_END_USER_ACCOUNT Procedure

Makes expired end users accounts and the associated passwords usable, enabling a end user to log in to a workspace.

Syntax

```
APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT (
    p_user_name IN VARCHAR2
    );
```

Parameters

Table 1–82 describes the parameters available in the UNEXPIRE_END_USER_ACCOUNT procedure.

Table 1–82 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 wwv_flow_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: " EXPIRE_END_USER_ACCOUNT Parameters" on page 1-23

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 1–83 describes the parameters available in the UNEXPIRE_WORKSPACE_ACCOUNT procedure.

Table 1–83 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 with respect to 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 wwv_flow_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" on page 1-24 and

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 1–84 describes the parameters available in the UNLOCK_ACCOUNT procedure.

Table 1–84 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 wwv_flow_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" on page 1-65 and "GET_ACCOUNT_LOCKED_STATUS Function" on page 1-32

URL_ENCODE Function

This function encodes (into hexadecimal) all special characters that include spaces, question marks, and ampersands.

Syntax

```
APEX_UTIL.URL_ENCODE (
    p_url   IN    VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

Table 1–85 describes the parameters available in the URL_ENCODE function.

Table 1–85 URL_ENCODE Parameters

Parameter	Description
p_url	The string to be encoded

Example

```
DECLARE
      l_url  VARCHAR2(255);
BEGIN
      l_url := APEX_UTIL.URL_ENCODE('http://www.myurl.com?id=1&cat=foo');
END;
```

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 1–86 describes the parameters available in the WORKSPACE_ACCOUNT_DAYS_LEFT procedure.

Table 1–86 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 wwv_flow_users) LOOP
    l_days_left := APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT(p_user_name =>
c1.user_name) THEN
    htp.p('Workspace Account:'||c1.user_name||' will expire in '||l_days_left||'
days.');
  END LOOP;
END;
```

2

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 in order to use APEX_MAIL.

> **See Also:** *Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_SMTP package

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.

This section contains the following topics:

- SEND Procedure
- ADD_ATTACHMENT Procedure
- PUSH_QUEUE Procedure
- APEX_MAIL_ATTACHMENTS View

Note: The most efficient approach to sending email is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

> **See Also:** "Sending Email from an Application" in *Oracle Application Express Application Builder User's Guide*

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 will result 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 will not display. 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,
    p_subj                      IN    VARCHAR2 DEFAULT)
    p_cc                        IN    VARCHAR2 DEFAULT)
    p_bcc                       IN    VARCHAR2 DEFAULT);
    p_replyto                   IN    VARCHAR2 DEFAULT);
RETURN NUMBER;
```

Parameters

Table 2–1 describes the parameters available in the SEND procedure.

Table 2–1 SEND Parameters

Parameter	Description
p_to	Valid email address to which the email will be sent (required). For multiple email addresses, use a comma-separated list
p_from	Email address from which the email will be sent (required). This email address must be a valid address. Otherwise, the message will not be 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 will send these messages, but the automatic replies will 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 APEX 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 HTMLDB Dev
Team</span><br />'||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_body_html => l_body_html,
     p_subj      => 'APEX_MAIL Package - HTML formatted message');
END;
/
```

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 2–2 describes the parameters available in the ADD_ATTACHMENT procedure.

Table 2–2 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 e-mail body.
p_attachment	A BLOB variable containing the binary content to be attached to the e-mail message.
p_filename	The filename associated with the e-mail attachment.
p_mime_type	A valid MIME type (or Internet media type) to associate with the e-mail attachment.

Examples

The following example demonstrates how to access files stored in APEX_APPLICATION_FILES and add them an 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,
                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;
/
```

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. Keep in mind, the most efficient approach to sending email is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

> **See Also:** "Sending an Email from an Application" in *Oracle Application Express Application Builder User's Guide*

Syntax

```
APEX_MAIL.PUSH_QUEUE(
    p_smtp_hostname             IN    VARCHAR2 DEFAULT,
    p_smtp_portno               IN    NUMBER   DEFAULT;
```

Parameters

Table 2–3 describes the parameters available in the PUSH_QUEUE procedure.

Table 2–3 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 e-mail.

> **See Also:** "Configuring Email Settings" in *Oracle Application Express Administration Guide*

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 Email from an Application" in *Oracle Application Express Application Builder User's Guide*

APEX_MAIL_ATTACHMENTS View

You can use the APEX_MAIL_ATTACHMENTS view in conjunction with the existing APEX_MAIL_QUEUE to access email attachments associated with email messages in the Oracle Application Express mail queue.

See Also: "PUSH_QUEUE Procedure" on page 2-7

Example

The following example demonstrates how to access files stored in APEX_APPLICATION_FILES and add to an e-mail 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;
/
```

See Also: "Sending Email from an Application" in *Oracle Application Express Application Builder User's Guide*

3

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.

This section contains the following topics:

CHECKBOX Function

This function creates check boxes.

Syntax

```
APEX_ITEM.CHECKBOX(
    p_idx                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT,
    p_attributes                IN    VARCHAR2 DEFAULT,
    p_checked_values            IN    VARCHAR2 DEFAULT,
    p_checked_values_delimiter  IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–1 describes the parameters available in the CHECKBOX function.

Table 3–1 CHECKBOX Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable will be 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 HTML tag attributes (such as disabled)
p_checked_values	Values to be checked by default
p_checked_values_delimiter	Delimits the values in the previous parameter, p_checked_values

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.CHECKBOX(1,empno,'CHECKED') " ",
       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.CHECKBOX(1,empno) " ",
       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.CHECKBOX(1,empno,DECODE(deptno,10,'CHECKED',NULL)) " ",
       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.CHECKBOX(1,deptno,NULL,'10:20',':') " ",
       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 utilizes the following logic:

```
SELECT APEX_ITEM.CHECKBOX(1,empno) " ",
       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;
```

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,
    p_date_format  IN    DATE DEFAULT,
    p_size         IN    NUMBER DEFAULT,
    p_maxlength    IN    NUMBER DEFAULT,
    p_attributes   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–2 describes the parameters available in the DATE_POPUP function.

Table 3–2 DATE_POPUP Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable will be 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 will be 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

> **See Also:** *Oracle Database SQL Language Reference* for information about the TO_CHAR or TO_DATE functions

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
```

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 3–3 describes the parameters available in the `DISPLAY_AND_SAVE` function.

Table 3–3 DISPLAY_AND_SAVE Parameters

Parameter	Description
`p_idx`	Number that determines which `APEX_APPLICATION` global variable will be 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 `<input>` tag
`p_item_label`	Label of the text field 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
```

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 3–4 describes the parameters available in the `HIDDEN` function.

Table 3–4 HIDDEN Parameters

Parameter	Description
p_idx"	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated **See Also:** "APEX_APPLICATION" on page 4-1
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	Label of the text field 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.

MD5_CHECKSUM Function

This function passes values to APEX_ITEM.MULTI_ROW_UPDATE and is used for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

Syntax

```
APEX_ITEM.MD5_CHECKSUM(
    p_value01   IN    VARCHAR2 DEFAULT,
    p_value02   IN    VARCHAR2 DEFAULT,
    p_value03   IN    VARCHAR2 DEFAULT,
    ...
    p_value50   IN    VARCHAR2 DEFAULT,
    p_col_sep   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–5 describes the parameters available in the MD5_CHECKSUM function.

Table 3–5 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 (|)

Example

```
SELECT APEX_ITEM.MD5_CHECKSUM(ename,job,sal)
FROM emp
```

MD5_HIDDEN Function

This function is used 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 and includes 50 inputs. APEX_ITEM.MD5_HIDDEN also produces an MD5 checksum using the Oracle database DBMS_OBFUSCATION_TOOLKIT:

```
UTL_RAW.CAST_TO_RAW(DBMS_OBFUSCATION_TOOLKIT.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,
    p_value02   IN    VARCHAR2 DEFAULT,
    p_value03   IN    VARCHAR2 DEFAULT,
    ...
    p_value50   IN    VARCHAR2 DEFAULT,
    p_col_sep   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–6 describes the parameters available in the MD5_HIDDEN function.

Table 3–6 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	Fifty available inputs. Parameters not supplied default to NULL
...	
p_value50	
p_col_sep	String used to separate p_value inputs. Defaults to the pipe symbol (|)

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 will be generated.

```
SELECT APEX_ITEM.MD5_HIDDEN(7,ename,job,sal), ename, job, sal FROM emp
```

MULTI_ROW_UPDATE Procedure

Use this procedure within a Multi Row Update process type. This procedure takes a string containing a multiple row update definition in the following format:

```
OWNER:TABLE:pk_column1,pk_idx:pk_column2,pk_idx2|col,idx:col,idx...
```

Syntax

```
APEX_ITEM.MULTI_ROW_UPDATE(
    p_mru_string    IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Example

To use this procedure indirectly within an application-level process, you need to create a query to generate a form of database data. The following example demonstrates how to create a multiple row update on the `emp` table.

```
SELECT
empno,
APEX_ITEM.HIDDEN(1,empno),
APEX_ITEM.HIDDEN(2,deptno),
APEX_ITEM.TEXT(3,ename),
APEX_ITEM.SELECT_LIST_FROM_QUERY(4,job,'SELECT DISTINCT job FROM emp'),
APEX_ITEM.TEXT(5,sal),
APEX_ITEM.TEXT(7,comm),
APEX_ITEM.MD5_CHECKSUM(ename,job,sal,comm),
deptno
FROM emp
WHERE deptno = 20
```

Note the call to `APEX_ITEM.MD5_CHECKSUM`, instead of `APEX_ITEM.MD5_HIDDEN`. Since `APEX_ITEM.MULTI_ROW_UPDATE` gets the checksum from `APEX_APPLICATION.G_FCS`, you need to call `APEX_ITEM.MD5_CHECKSUM` in order to populate `APEX_APPLICATION.G_FCS` when the page is submitted. Additionally, the columns in `APEX_ITEM.MD5_CHECKSUM` must be in the same order those in the `MULTI_ROW_UPDATE` process. These updates can then processed (or applied to the database) using an after submit page process of Multi Row Update in a string similar to the following:

```
SCOTT:emp:empno,1:deptno,2|ename,3:job,4:sal,5:comm,7:,:,:,:,
```

POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application list of values (LOV). Similar from 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,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–7 describes the some parameters in the POPUP_FROM_LOV function.

Table 3–7 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 Web site). 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 3–7 (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 will be escaped. This parameter is useful if you know your query will return 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.

```
SELECT APEX_ITEM.POPUP_FROM_LOV (1,deptno,'DEPT_LOV') dt
FROM emp
```

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,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–8 describes the parameters in the POPUP_FROM_QUERY function.

Table 3–8 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 Web site). 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 3–8 (Cont.) POPUP_FROM_QUERY 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 will be escaped. This parameter is useful if you know your query will return 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
```

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,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    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 3–9 describes the some parameters in the POPUPKEY_FROM_LOV function.

Table 3–9 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.
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 Web site). 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 3–9 (Cont.) POPUPKEY_FROM_LOV Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ <lt; for < ■ <gt; for > ■ <amp; for & This parameter is useful if you know your query will return 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.

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
```

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,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–10 describes the some parameters in the POPUPKEY_FROM_QUERY function.

Table 3–10 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 Web site). 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 3–10 (Cont.) POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ <lt; for < ■ <gt; for > ■ <amp; for & This parameter is useful if you know your query will return 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 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
```

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,
    p_selected_value   IN    VARCHAR2 DEFAULT,
    p_display          IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_onblur           IN    VARCHAR2 DEFAULT,
    p_onchange         IN    VARCHAR2 DEFAULT,
    p_onfocus          IN    VARCHAR2 DEFAULT,)
    RETURN VARCHAR2;
```

Parameters

Table 3–11 describes the parameters available in the `RADIOGROUP` function.

Table 3–11 RADIOGROUP Parameters

Parameter	Description
`p_idx`	Number that determines which `APEX_APPLICATION` global variable will be 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.
`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.

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
```

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,
    p_list_values   IN   VARCHAR2 DEFAULT,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT,
    p_show_extra    IN   VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–12 describes the parameters available in the SELECT_LIST function.

Table 3–12 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	Label of the select list.
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')
```

```
FROM emp
```

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.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–13 describes the parameters available in the SELECT_LIST_FROM_LOV function.

Table 3–13 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.
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 <input> tag.
p_item_label	Label of 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')
FROM emp
```

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. This enables you to use it in SQL queries where you need to handle a column value longer than 4000 characters.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV_XL(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT)
    RETURN CLOB;
```

Parameters

Table 3–14 describes the parameters available in the SELECT_LIST_FROM_LOV_XL function.

Table 3–14 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 <input> tag.
p_item_label	Label of 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')
FROM emp
```

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,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT,
    p_show_null     IN    VARCHAR2 DEFAULT,
    p_null_value    IN    VARCHAR2 DEFAULT,
    p_null_text     IN    VARCHAR2 DEFAULT,
    p_item_id       IN    VARCHAR2 DEFAULT,
    p_item_label    IN    VARCHAR2 DEFAULT,
    p_show_extra    IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 3–15 describes the parameters available in the SELECT_LIST_FROM_QUERY function.

Table 3–15 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.
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	Label of the select list.
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')
FROM emp
```

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. 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,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT,
    p_show_null     IN    VARCHAR2 DEFAULT,
    p_null_value    IN    VARCHAR2 DEFAULT,
    p_null_text     IN    VARCHAR2 DEFAULT,
    p_item_id       IN    VARCHAR2 DEFAULT,
    p_item_label    IN    VARCHAR2 DEFAULT,
    p_show_extra    IN    VARCHAR2 DEFAULT)
    RETURN CLOB;
```

Parameters

Table 3–16 describes the parameters available in the SELECT_LIST_FROM_QUERY_XL function.

Table 3–16 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.
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	Label of the select list.
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')
FROM emp
```

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,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–17 describes the parameters available in the `TEXTAREA` function.

Table 3–17 TEXTAREA Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated. **See Also:** **"APEX_APPLICATION"** on page 4-1
`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 `<input>` tag.
`p_item_label`	Label of the text area 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
```

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)
```

Parameters

Table 3–18 describes the parameters available in the `TEXT` function.

Table 3–18 TEXT Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated. **See Also:** "APEX_APPLICATION" on page 4-1
`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`	Label of the text field 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
```

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 3–19 describes the parameters available in the TEXT_FROM_LOV function.

Table 3–19 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
```

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 3–20 describes the parameters available in the `TEXT_FROM_LOV_QUERY` function.

Table 3–20 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`
`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
```

4

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 a number of global variables. Table 4–1 describes the global variables available in the APEX_APPLICATION package.

Table 4–1 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.

Topics in this section include:

- Referencing Arrays
- Referencing Values Within an On Submit Process
- Converting an Array to a Single Value

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.

If you need 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;
```

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 will 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 will only have an entry in the APEX_APPLICATION array if it is selected.

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. 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)
```

5

APEX_CUSTOM_AUTH

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

Topics in this section include:

APPLICATION_PAGE_ITEM_EXISTS Function

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

Syntax

```
FUNCTION APPLICATION_PAGE_ITEM_EXISTS(
    p_item_name   IN    VARCHAR2)
RETURN BOOLEAN;
```

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)

> **See Also:** "Editing Page Attributes" in *Oracle Application Express Application Builder User's Guide*.

Syntax

```
FUNCTION CURRENT_PAGE_IS_PUBLIC
RETURN BOOLEAN;
```

DEFINE_USER_SESSION Procedure

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

Syntax

```
PROCEDURE DEFINE_USER_SESSION(
    p_user         IN    VARCHAR2)
    p_session_id   IN    NUMBER);
```

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 Application Builder by viewing the authentication scheme 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);
```

Parameters

Table 5–1 describes the parameters available in the `GET_COOKIE_PROPS` procedure.

Table 5–1 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.

Example

```
DECLARE
    l_cookie_name   varchar2(256);
    l_cookie_path   varchar2(256);
    l_cookie_domain varchar2(256);
BEGIN
APEX_CUSTOM_AUTH.GET_COOKIE_PROPS (
    p _cookie_name   => l_cookie_name,
    p _cookie_path   => l_cookie_path,
    p _cookie_domain => l_cookie_domain);
END;
```

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 Application Builder by viewing the authentication scheme attributes.

Syntax

```
APEX_CUSTOM_AUTH.GET_LDAP_PROPS(
 p_ldap_host                OUT VARCHAR2,
 p_ldap_port                OUT NUMBER,
 p_ldap_dn                  OUT VARCHAR2,
 p_ldap_edit_function       OUT VARCHAR2);
```

Parameters

Table 5–2 describes the parameters available in the GET_LDAP_PROPS procedure.

Table 5–2 GET_LDAP_PROPS Parameters

Parameter	Description
p_ldap_host	LDAP host name.
p_ldap_port	LDAP port number.
p_ldap_dn	LDAP DN string.
p_ldap_edit_function	LDAP edit function name.

Example

```
DECLARE
    l_ldap_host          varchar2(256);
    l_ldap_port          number;
    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_ldap_dn         => l_ldap_dn,'
    p_ldap_edit_function => l_ldap_edit_function);
END;
```

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

```
FUNCTION GET_NEXT_SESSION_ID
RETURN NUMBER;
```

GET_SESSION_ID_FROM_COOKIE Function

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

Syntax

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

Example

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

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

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

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

```
FUNCTION GET_SECURITY_GROUP_ID
RETURN NUMBER;
```

GET_SESSION_ID Function

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

Syntax

```
PROCEDURE GET_SESSION_ID
RETURN NUMBER;
```

GET_USER Function

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

Syntax

```
FUNCTION GET_USER
RETURN VARCHAR2;
```

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

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := APEX_CUSTOM_AUTH.IS_SESSION_VALID;
END;
```

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,
 p_password                 IN  VARCHAR2,
 p_session_id               IN  VARCHAR2,
 p_app_page                 IN  VARCHAR2,
 p_entry_point              IN  VARCHAR2,
 p_preserve_case            IN  BOOLEAN);
```

Parameter

Table 5–3 describes the parameters available in the `LOGIN` procedure.

Table 5–3 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

```
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.

LOGOUT Procedure

This procedure effects 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,
 p_next_app_page_sess          IN VARCHAR2,
 p_next_url                    IN VARCHAR2);
```

Parameter

Table 5–4 describes the parameters available in the `LOGOUT` procedure.

Table 5–4 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

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

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,
 p_session_id               IN  VARCHAR2,
 p_app_page                 IN  VARCHAR2,
 p_preserve_case            IN  BOOLEAN);
```

Parameter

Table 5–5 describes the parameters available in the `POST_LOGIN` procedure.

Table 5–5 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

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

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. returns false if the result is a negative number.

Syntax

```
FUNCTION SESSION_ID_EXISTS
RETURN BOOLEAN;
```

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := APEX_CUSTOM_AUTH.SESSION_ID_EXISTS;
END;
```

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

```
PROCEDURE SET_USER(
    p_user   IN    VARCHAR2)
```

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

```
PROCEDURE SET_SESSION_ID(
    p_session_id    IN    NUMBER)
```

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

```
PROCEDURE SET_SESSION_ID_TO_NEXT_VALUE;
```

6

APEX_LDAP

You can use `APEX_LDAP` to perform various operations related to Lightweight Directory Access Protocol (LDAP) authentication.

Topics in this section include:

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

```
FUNCTION 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)
RETURN BOOLEAN;
```

Parameters

Table 6–1 describes the parameters available in the AUTHENTICATE function.

Table 6–1 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.
p_port	LDAP server port number.

Example

```
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;
```

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

```
FUNCTION IS_MEMBER(
    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_group        IN VARCHAR2,
    p_group_base   IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 6–2 describes the parameters available in the IS_MEMBER function.

Table 6–2 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_group	Name of the group to be search for membership.
p_group_base	The base from which the search should be started.

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

```
FUNCTION 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)
RETURN wwv_flow_global.vc_arr2;
```

Parameters

Table 6–3 describes the parameters available in the MEMBER_OF function.

Table 6–3 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.

MEMBER_OF2 Function

The MEMBER_OF2 function returns an VARCHAR2 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

```
FUNCTION 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)
RETURN VARCHAR2;
```

Parameters

Table 6–4 describes the parameters available in the MEMBER_OF2 function.

Table 6–4 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.

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

```
PROCEDURE 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_attributes        IN  wwv_flow_global.vc_arr2,
    p_attribute_values  OUT wwv_flow_global.vc_arr2);
```

Parameters

Table 6–5 describes the parameters available in the GET_USER_ATTRIBUTES procedure.

Table 6–5 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_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.

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

```
PROCEDURE GET_ALL_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_attributes        OUT  wwv_flow_global.vc_arr2,
    p_attribute_values  OUT wwv_flow_global.vc_arr2);
```

Parameters

Table 6–6 describes the parameters available in the GET_ALL_USER_ATTRIBUTES procedure.

Table 6–6 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.
p_host	LDAP server host name.
p_port	LDAP server port number.
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.

7

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, wallet settings, report printing settings and to manage schema to workspace mappings. APEX_INSTANCE_ADMIN can be executed by the SYS, SYSTEM, and FLOWS_030100 database users as well as any database user granted the role APEX_ADMINISTRATOR_ROLE.

Topics in this section include:

SET_PARAMETER Procedure

The SET_PARAMETER procedure sets a parameter used in administering a runtime environment.

Syntax

```
APEX_INSTANCE_ADMIN.SET_PARAMETER(
    p_parameter     IN VARCHAR2,
    p_value         IN VARCHAR2);
```

Parameters

Table 7–1 describes the parameters available in the SET_PARAMETER procedure.

Table 7–1 SET_PARAMETER Parameters

Parameter	Description
p_parameter	The instance parameter to be set.
p_value	The value of the parameter. See "Available Parameter Values" on page 7-4.

Example

```
BEGIN
APEX_INSTANCE_ADMIN.SET_PARAMETER
('SMTP_HOST','mail.mycompany.com');
END;
```

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 7–2 describes the parameters available in the GET_PARAMETER function.

Table 7–2 GET_PARAMETER Parameters

Parameter	Description
p_parameter	The instance parameter to be retrieved. See "Available Parameter Values" on page 7-4.

Example

```
DECLARE L_VAL VARCHAR2(4000);
BEGIN
    L_VAL :=
APEX_INSTANCE_ADMIN.GET_PARAMETER('SMTP_HOST');
    HTP.P('The SMTP Host Setting Is: '||L_VAL);
END;
```

Available Parameter Values

Table 7–3 lists all the available parameter values you can set within the APEX_INSTANCE_ADMIN package, including parameters for email, wallet, and reporting printing.

Table 7–3 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
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.
PRINT_BIB_LICENSED	Specify either standard support or advanced support. Advanced support requires an Oracle BI Publisher license. Valid values include: ■ STANDARD ■ ADVANCED
PRINT_SVR_PROTOCOL	Valid values include: ■ http ■ https
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_SCRIPT	Defines the script that is the print server engine, for example: /xmlpserver/convert

See Also: "Configuring Email in a Runtime Environment", "Configuring a Wallet in a Runtime Environment", "Configuring Report Printing Settings in a Runtime Environment" in *Oracle Application Express Administration Guide*.

GET_SCHEMAS Function

The GET_SCHEMAS function retrieves a list of schemas that are mapped to a given workspace.

Syntax

```
APEX_INSTANCE_ADMIN.GET_SCHEMAS(
    p_workspace     IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 7–4 describes the parameters available in the GET_SCHEMAS function.

Table 7–4 GET_SCHEMAS Parameters

Parameter	Description
p_workspace	The name of the workspace from which to retrieve the schema list.

Example

```
DECLARE
    L_VAL VARCHAR2(4000);
BEGIN
    L_VAL :=
APEX_INSTANCE_ADMIN.GET_SCHEMAS('MY_WORKSPACE');
    HTP.P('The schemas for my workspace: '||L_VAL);
END;
```

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 7–5 describes the parameters available in the ADD_SCHEMA procedure.

Table 7–5 ADD_SCHEMA Parameters

Parameter	Description
p_workspace	The name of the workspace to which the schema mapping will be added.
p_schema	The schema to add to the schema to workspace mapping.

Example

```
BEGIN
APEX_INSTANCE_ADMIN.ADD_SCHEMA
('MY_WORKSPACE','FRANK');
END;
```

REMOVE_SCHEMA Procedure

This REMOVE_SCHEMA procedure removes a schema from a workspace to schema mapping.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA(
    p_workspace     IN VARCHAR2
    p_schema        IN VARCHAR2);
```

Parameters

Table 7–6 describes the parameters available in the REMOVE_SCHEMA procedure.

Table 7–6 REMOVE_SCHEMA Parameters

Parameter	Description
p_workspace	The name of the workspace from which the schema mapping will be removed.
p_schema	The schema to remove from the schema to workspace mapping.

Example

```
BEGIN
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA
('MY_WORKSPACE','FRANK');
END;
```

REMOVE_WORKSPACE Procedure

The REMOVE_WORKSPACE procedure removes a workspace from a an Application Express instance.

Syntax

```
PROCEDURE REMOVE_WORKSPACE(
    p_workspace         IN VARCHAR2
    p_drop_users        IN VARCHAR2 DEFAULT 'N',
    p_drop_tablespaces  IN VARCHAR2 DEFAULT 'N' );
```

Parameters

Table 7–7 describes the parameters available in the REMOVE_WORKSPACE procedure.

Table 7–7 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

```
BEGIN
APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE
('MY_WORKSPACE','Y','Y');
END;
```

ADD_WORKSPACE Procedure

The ADD_WORKSPACE procedure adds a workspace to an Application Express Instance.

Syntax

```
PROCEDURE REMOVE_WORKSPACE(
    p_workspace_id        IN NUMBER DEFAULT NULL,
    p_workspace           IN VARCHAR2,
    p_primary_schema      IN VARCHAR2,
    p_additional_schemas  IN VARCHAR2 );
```

Parameters

Table 7–8 describes the parameters available in the ADD_WORKSPACE procedure.

Table 7–8 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 will be assigned.
p_workspace	The name of the workspace to be added.
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.

Example

```
BEGIN
APEX_INSTANCE_ADMIN.ADD_WORKSPACE
(8675309,'MY_WORKSPACE','SCOTT','HR:OE');
END;
```

8

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.

Topics in this section include:

> **See Also:** "Managing User Interface Defaults" in *Oracle Application Express Application Builder User's Guide*

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 8–1 describes the parameters available in the UPD_FORM_REGION_TITLE procedure.

Table 8–1 APEX_UI_DEFAULT_UPDATE Parameters

Parameter	Description
p_table_name	Table name
p_form_region_title	Desired form region title

Example

```
APEX_UI_DEFAULT_UPDATE.UPD_FORM_REGION_TITLE (
    p_table_name         => 'DEPT',
    p_form_region_title  => 'Deptartment Details');
```

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 8–2 describes the parameters available in the UPD_REPORT_REGION_TITLE procedure.

Table 8–2 UPD_REPORT_REGION_TITLE Parameters

Parameter	Description
p_table_name	Table name
p_report_region_title	Desired report region title

Example

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_REGION_TITLE (
    p_table_name            => 'DEPT',
    p_report_region_title   => 'Deptartments');
```

UPD_ITEM_LABEL Procedure

The UPD_ITEM_LABEL procedure sets the label used for items. 'This user interface default will be used when you create a form based on the specified table and include a specific column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_LABEL (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_label                 IN VARCHAR2 DEFAULT NULL
   );
```

Parameters

Table 8–3 describes the parameters available in the UPD_ITEM_LABEL procedure.

Table 8–3 UPD_ITEM_LABEL Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_label	Desired item label

Example

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_LABEL(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_label => 'Department Number');
```

UPD_ITEM_HELP Procedure

The UPD_ITEM_HELP procedure updates the help text for the specified table and column. This user interface default will be 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 8–4 describes the parameters available in the UPD_ITEM_HELP procedure.

Table 8–4 UPD_ITEM_HELP Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_help_text	Desired help text

Example

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_help_text => 'The number assigned to the department.');
```

UPD_DISPLAY_IN_FORM Procedure

The UPD_DISPLAY_IN_FORM procedure sets the display in form user interface defaults. 'This user interface default will be used by wizards when you select to create a form based upon the table. It controls whether the column will be 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 DEFAULT NULL
   );
```

Parameters

Table 8–5 describes the parameters available in the UPD_DISPLAY_IN_FORM procedure.

Table 8–5 UPD_DISPLAY_IN_FORM Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_in_form	Determines whether or not to display in the form by default, valid values are Y and N

Example

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_display_in_form => 'N');
```

UPD_DISPLAY_IN_REPORT Procedure

The UPD_DISPLAY_IN_REPORT procedure sets the display in report user interface default. This user interface default will be used by wizards when you select to create a report based upon the table and controls whether the column will be 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 8–6 describes the parameters available in the UPD_DISPLAY_IN_REPORT procedure.

Table 8–6 UPD_DISPLAY_IN_REPORT Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_in_report	Determines whether or not to display in the report by default, valid values are Y and N

Example

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_REPORT(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_display_in_report => 'N');
```

UPD_ITEM_DISPLAY_WIDTH Procedure

'The UPD_ITEM_DISPLAY_WIDTH procedure sets the item display width user interface default. This user interface default will be used by wizards when you select to create a form based upon the table and include the specified column.n.

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 8–7 describes the parameters available in the UPD_ITEM_DISPLAY_WIDTH procedure.

Table 8–7 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

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_WIDTH(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_display_width => 5);
```

UPD_ITEM_DISPLAY_HEIGHT Procedure

'The UPD_ITEM_DISPLAY_HEIGHT procedure sets the item display height user interface default. This user interface default will be 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 will be 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 8–8 describes the parameters available in the UPD_ITEM_DISPLAY_HEIGHT procedure.

Table 8–8 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

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT(
   p_table_name => 'DEPT',
   p_column_name => 'DNAME',
   p_display_height => 3);
```

UPD_REPORT_ALIGNMENT Procedure

'The UPD_REPORT_ALIGNMENT procedure sets the report alignment user interface default. This user interface default will be 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 8–9 describes the parameters available in the `UPD_REPORT_ALIGNMENT` procedure.

Table 8–9 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

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_ALIGNMENT(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_report_alignment => 'R');
```

UPD_ITEM_FORMAT_MASK Procedure

'The UPD_ITEM_FORMAT_MASK procedure sets the item format mask user interface default. This user interface default will be 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 8–10 describes the parameters available in the UPD_ITEM_FORMAT_MASK procedure.

Table 8–10 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

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_FORMAT_MASK(
   p_table_name => 'EMP',
   p_column_name => 'HIREDATE',
   p_format_mask=> 'DD-MON-YYYY');
```

UPD_REPORT_FORMAT_MASK Procedure

'The UPD_REPORT_FORMAT_MASK procedure sets the report format mask user interface default. This user interface default will be 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 8–11 describes the parameters available in the UPD_REPORT_FORMAT_MASK procedure.

Table 8–11 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

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_FORMAT_MASK(
   p_table_name => 'EMP',
   p_column_name => 'HIREDATE',
   p_format_mask=> 'DD-MON-YYYY');
```

9

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 XML HTTP Asynchronous JavaScript and XML (AJAX) requests.

Topics in this section include:

- $x_ByClass(pClass, pNd, pTag)
- $x_ShowAllByClass(pNd, pClass, pTag)
- $x_ShowChildren(pNd)
- $x_HideChildren(pNd)
- $x_disableItem(pNd, pTest)
- $f_get_emptys(pNd, pClassFail, pClass)
- $v_Array(pNd)
- $f_ReturnChecked(pNd)
- $d_ClearAndHide(pNd)
- $f_SelectedOptions(pNd)
- $f_SelectValue(pNd)
- $u_ArrayToString(pArray, pDelim)
- $x_CheckImageSrc(pId,pSearch)
- $v_CheckValueAgainst(pThis, pValue)
- $f_Hide_On_Value_Item(pThis, pThat, pValue)
- $f_Show_On_Value_Item(pThis, pThat, pValue)
- $f_Hide_On_Value_Item_Row(pThis, pThat, pValue)
- $f_Show_On_Value_Item_Row(pThis, pThat, pValue)
- $f_DisableOnValue(pThis, pValue, pThat)
- $x_ClassByClass(pNd, pClass, pTag, pClass2)
- $f_ValuesToArray(pThis, pClass, pTag)
- $x_FormItems(pNd, pType)
- $f_CheckAll(pThis, pCheck, pArray)
- $f_CheckFirstColumn(pNd)
- $v_PopupReturn(pValue, pThat)
- $x_ToggleWithImage(pThis,pNd)
- $x_SwitchImageSrc(pNd, pSearch, pReplace)
- $x_CheckImageSrc(pNd, pSearch)
- $u_SubString(pText,pMatch)
- html_RemoveAllChildren(pNd)
- $v_IsEmpty(pThis)
- html_SetSelectValue(pId,pValue)
- addLoadEvent(pFunction)
- $f_Swap(pThis,pThat)
- submitEnter(pNd,e)
- $f_SetValueSequence(pArray,pMultiple)
- $dom_AddTag(pThis, pTag, pText)

- $tr_AddTD(pThis,pText)
- $dom_AddInput(pThis,pType,pId,pName,pValue)
- $dom_MakeParent(p_Node,p_Parent)
- $x_RowHighlight(pThis, pColor)
- $x_RowHighlightOff(pThis)
- $v_Upper(pNd)
- $v_Upper(pNd)
- $d_Find(pThis,pString,pTags,pClass)
- returnInput(p_R, p_D)
- setReturn(p_R,p_D)
- $f_First_field(pNd)
- GetCookie (pName)
- SetCookie (pName,pValue)

$x(pNd)

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.

Return Value

```
(DOM Node | false)
```

Parameters

```
pNd (DOM Node | string ID)
```

$v(pNd)

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)
```

$s(pNd, pValue)

Given a DOM node or string ID (pNd), this function sets the Application Express item value taking into account what type of item it is.

Parameters

```
pNd (DOM Node | string ID)
pValue  (String | Array)
```

$u_Carray(pNd)

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.

Return Value

```
pNd (DOM Node | string ID | Array)
```

Parameters

```
Array
```

$u_Narray(pNd)

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 will be returned otherwise the array will be returned. Used for creating DOM based functionality that can accept a single or multiple DOM nodes.

Return Value

```
Array (DOM Node | string ID | Array)
```

Parameters

```
Array or first value
```

$nvl(pTest, pDefault)

If `pTest` is empty or false return `pDefault` otherwise return `pTest`.

Return Value

`(string | Array)`

Parameters

`pTest  (String | Array)`
`pDefault (String | Array)`

doSubmit(pRequest)

Submits the page setting the Application Express Request value (`pRequest`).

Parameters

`pRequest (String)`

confirmDelete(pMessage, pRequest)

Displays a confirmation showing a message (`pMessage`) and depending on user's choice, submits a page setting request value (`pRequest`) or cancels page submit.

Parameters

```
pMessage (string)
pRequest (string)
```

$x_Style(pNd, pStyle, pString)

Sets a specific style property (`pStyle`) to given value (`pString`) of a DOM node or DOM node Array (`pNd`).

Return Value

```
(DOM node | DOM Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array )
pStyle (String)
pString (String)
```

$x_Hide(pNd)

Hides a DOM node or array of DOM nodes (pNd).

Return Value

(DOM node | Array)

Parameters

pNd (DOM node | string ID | DOM node Array)

$x_Show(pNd)

Shows a DOM node or array of DOM nodes (pNd).

Return Value

(DOM node | Array)

Parameters

pNd (DOM node | string ID | DOM node Array)

$x_Toggle(pNd)

Toggles a DOM node or array of DOM nodes (pNd).

Return Value

```
(DOM node | Array)
```

Parameters

```
pNd (DOM node | string ID | Array)
```

$x_Remove(pNd)

Removes a DOM node or array of DOM nodes.

Return Value

```
(DOM Node | Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array)
```

$x_Value(pNd,pValue)

Sets the value (`pValue`) of a DOM node or array of DOM nodes (`pNd`).

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | string ID | DOM node Array)`
`pValue (String)`

$x_UpTill(pNd, pToTag)

Starting from a DOM node (pNd), this function cascades up the DOM tree until the tag of node name (pToTag) is found.

Return Value

```
(DOM Node | false)
```

Parameters

```
pNd  (DOM Node | string ID)
String (pToTag)
String (pToClass )
```

$x_ItemRow(pNd,pFunc)

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.

Return Value

Not applicable.

Parameters

pNd (DOM Node | string ID | Dom node Array)
pFunc ['TOGGLE','SHOW','HIDE'] (String)

$x_HideItemRow(pNd)

Given a page item name, this function hides the entire row that holds the item. In most cases, this will be the item and its label.

Return Value

Not applicable.

Parameters

pNd (DOM Node | string ID | DON node Array)

$x_ShowItemRow(pNd)

Given a page item name, this function shows the entire row that holds the item. In most cases, this will be the item and its label.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM note Array)

$x_ToggleItemRow(pNd)

Given a page item name (pNd), this function toggles the entire row that holds the item. In most cases, this will be the item and its label.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM node ray)

$x_HideAllExcept(pNd,pNdArray)

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`.

Return Value

```
(DOM node | DOM Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array)
pNdArray (DOM node | String | Array)
```

$x_HideSiblings(pNd)

Hides all sibling nodes of given `pNd`.

Return Value

`(DOM node)`

Parameters

`pNd (DOM node | string ID )`

$x_ShowSiblings(pNd)

Shows all sibling DOM nodes of given DOM nodes (`pNd`).

Return Value

`(DOM node)`

Parameters

`pNd (DOM node | string ID )`

$x_Class(pNd,pClass)

Sets a DOM node or array of DOM nodes to a single class name.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM node Array)
pClass (String)

$x_SetSiblingsClass(pNd, pClass, pNdClass)

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`.

Return Value

```
(DOM node | false)
```

Parameters

```
pNd (DOM Nnde | string ID)
pClass (String)
pThisClass (String)
```

$x_ByClass(pClass, pNd, pTag)

Returns an array of DOM nodes by a given class name (`pClass`). If the `pNd` parameter is provided, then the returned elements will be all be children of that DOM node. Including the `pTag` parameter further narrows the list to just return nodes of that tag type.

Return Value

```
(Array)
```

Parameters

```
pClass (String)
pNd  (DOM node | string ID)
pTag (String)
```

$x_ShowAllByClass(pNd, pClass, pTag)

Show all the DOM node children of a DOM node (`pNd`) that have a specific class (`pClass`) and tag (`pTag`).

Return Value

`Not applicable.`

Parameters

```
pNd (DOM node | string ID)
pClass (String)
pTag (String)
```

$x_ShowChildren(pNd)

Show all DOM node children of a DOM node (pNd).

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID)

$x_HideChildren(pNd)

Hide all DOM node children of a DOM node (`pNd`).

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | string ID)`

$x_disableItem(pNd, pTest)

Disables or enables an item or array of items based on (`pTest`).

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | string ID | DOM node array)`
`a (true | false)`

$f_get_emptys(pNd, pClassFail, pClass)

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`.

Return Value

```
false, Array  Array of all items that are empty (false | Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array)
Sting (pClassFail)
Sting (pClass)
```

$v_Array(pNd)

Returns an item value as an array. Useful for multiselects and checkboxes.

Return Value

(Array)

Parameters

pId (DOM Node | string ID)

$f_ReturnChecked(pNd)

Returns an item value as an array. Useful for radio items and check boxes.

Return Value

```
(Array)
```

Parameters

```
pId (DOM node | string ID)
```

$d_ClearAndHide(pNd)

Clears the content of an DOM node or array of DOM nodes and hides them.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM node array)

$f_SelectedOptions(pNd)

Returns the DOM nodes of the selected options of a select item (pNd).

Return Value

(DOM Array)

Parameters

pNd (DOM node | string ID)

$f_SelectValue(pNd)

Returns the values of the selected options of a select item (`pNd`).

Return Value

```
(DOM Array | String)
```

Parameters

```
pNd (DOM node | string ID)
```

$u_ArrayToString(pArray, pDelim)

Given an array (`pArray`) return a string with the values of the array delimited with a given delimiter character (`pDelim`).

Return Value

`Not applicable.`

Parameters

`pArray (pArray)`
`pDelim (String)`

$x_CheckImageSrc(pId,pSearch)

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.

Return Value

```
(true | false)
```

Parameters

```
pId (DOM Node | String)
pSearch (pSearch)
```

$v_CheckValueAgainst(pThis, pValue)

Checks an page item's (`pThis`) value against a set of values (`pValue`). This function returns true if any value matches.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pValue (Number | String | Array)
```

$f_Hide_On_Value_Item(pThis, pThat, pValue)

Checks an 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.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_Show_On_Value_Item(pThis, pThat, pValue)

Checks an 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.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_Hide_On_Value_Item_Row(pThis, pThat, pValue)

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.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_Show_On_Value_Item_Row(pThis, pThat, pValue)

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.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_DisableOnValue(pThis, pValue, pThat)

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.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pValue (String)
pThat  (DOM node | string ID | DOM node Array )
```

$x_ClassByClass(pNd, pClass, pTag, pClass2)

Sets a class attribute of an array of nodes that are selected by class.

Return Value

```
(DOM node | DOM node Array)
```

Parameters

```
pNd (DOM node | string ID)
pClass (String)
pTag (String)
pClass2 (String)
```

$f_ValuesToArray(pThis, pClass, pTag)

Collects the values of form items contained within DOM node (`pThis`) of class attribute (`pClass`) and nodeName (`pTag`) and returns an array.

Return Value

No applicable.

Parameters

pThis (DOM node | string ID)
pCLass (String)
pTag (String)

$x_FormItems(pNd, pType)

Returns all form input items contained in a DOM node (`pThis`) of a certain type (`pType`).

Return Value

```
DOM node Array
```

Parameters

```
pNd (DOM node | string ID)
pType (String)
```

$f_CheckAll(pThis, pCheck, pArray)

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.

Return Value

`Not applicable.`

Parameters

```
pThis (DOM node | string ID)
pCheck (true | fales)
pArray (DOM node array)
```

$f_CheckFirstColumn(pNd)

This function sets all checkboxes located in the first column of a table based on the checked state of the calling checkbox (`pNd`), useful for tabular forms.

Return Value

`DOM node Array`

Parameters

`pNd (DOM node | String)`

$v_PopupReturn(pValue, pThat)

Sets the value of the item in the parent window (`pThat`), with (`pValue`) and then closes the popup window.

Return Value

Not applicable.

Parameters

pValue (string)
pThat (DOM node | string ID)

$x_ToggleWithImage(pThis,pNd)

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`) will be rewritten. The image src will have any plus substrings replaced with minus substrings or minus substrings will be replaced with plus substrings.

Return Value

```
(DOM Node)
```

Parameters

```
pThis (DOM Node | string ID)
pNd (DOM Nnde | string iD | DOM node Array)
```

$x_SwitchImageSrc(pNd, pSearch, pReplace)

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`).

Return Value

```
(DOM node | false)
```

Parameters

```
pNd (DOM node | string ID)
pSearch (String)
pReplace (String)
```

$x_CheckImageSrc(pNd, pSearch)

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.

Return Value

```
(true | fales)
```

Parameters

```
pNd  (DOM node | string ID)
pSearch (String)
```

$u_SubString(pText,pMatch)

Returns a true or false if a string (`pText`) contains a substring (`pMatch`).

Return Value

`(true | false)`

Parameters

`pText (String)`
`pMatch (String)`

html_RemoveAllChildren(pNd)

Use DOM methods to remove all DOM children of DOM node (pND).

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID)

$v_IsEmpty(pThis)

Returns true or false if a form element is empty, this will consider any whitespace including a space, a tab, a form-feed, as empty.

Return Value

```
[true | false]
```

Parameters

```
pThis (DOM Node | String)
```

html_SetSelectValue(pId,pValue)

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).

Return Value

`Not applicable.`

Parameters

`pId (DOM node | String)`
`pValue (String)`

addLoadEvent(pFunction)

Adds an onload function (`func`) without overwriting any previously specified onload functions.

Return Value

`Not applicable.`

Parameters

`pFunction (Javascript Function)`

$f_Swap(pThis,pThat)

Swaps the form values of two form elements (`pThis`,`pThat`).

Return Value

`Not applicable.`

Parameters

`pThis (DOM Node | String)`
`pThat (DOM Node | String)`

submitEnter(pNd,e)

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)"`

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | String | Array)`

$f_SetValueSequence(pArray,pMultiple)

Sets array of form item (`pArray`) to sequential number in multiples of (`pMultiple`).

Return Value

`Not applicable.`

Parameters

`pArray (Array)`
`pMultiple (Number)`

$dom_AddTag(pThis, pTag, pText)

Inserts the html element (`pTag`) as a child node of a DOM node (`pThis`) with the innerHTML set to (`pText`).

Return Value

`DOM node`

Parameters

`pThis (DOM node | string ID )`
`pTag (String)`
`pText (String)`

$tr_AddTD(pThis,pText)

Appends a table cell to a table row (`pThis`). And sets the content to (`pText`).

Return Value

```
(DOM node)
```

Parameters

```
pThis (DOM node | string ID)
pText (String)
```

$tr_AddTH(pThis,pText)

Appends a table cell to a table row (`pThis`). And sets the content to (`pText`).

Return Value

`DOM node`

Parameters

```
pThis (DOM node | string ID)
pTest (String)
```

$dom_AddInput(pThis,pType,pId,pName,pValue)

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`.

Return Value

```
(DOM node)
```

Parameters

```
pThis (DOM node | string ID)
pType (String)
pId (String)
pName (String)
pValue (String)
```

$dom_MakeParent(p_Node,p_Parent)

Takes a DOM node (p_Node) and makes it a child of DOM node (p_Parent) and then returns the DOM node (pNode).

Return Value

(DOM node)

Parameters

p_This (DOM node | string ID)
p_Parent (DOM node | string ID)

$x_RowHighlight(pThis, pColor)

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`.

Return Value

`Not applicable.`

Parameters

`pThis (DOM node | String)`
`pColor(String)`

$x_RowHighlightOff(pThis)

Give an table row Dom node (`pThis`), this function sets the background of all table cells to `NULL`.

Return Value

`Not applicable.`

Parameters

`pThis (DOM Element | String)`

$v_Upper(pNd)

Sets the value of a form item (pNd) to uppercase.

Return Value

Not applicable.

Parameters

pNd (DOM Node | String)

$d_Find(pThis,pString,pTags,pClass)

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 will be set to a block level element when set to visible.

Return Value

```
Not applicable.
```

Parameters

```
pThis (DOM node | String)
pString (String)
pTags (String
pClass (String)
```

returnInput(p_R, p_D)

Sets DOM node in the global variables `returnInput (p_R)` and `returnDisplay (p_D)` for use in populating items from popups.

Return Value

`Not applicable.`

Parameters

`p_R (DOM node | String)`
`p_R (DOM node | String)`

setReturn(p_R,p_D)

Sets DOM items in the global variables `returnInput` (`p_R`) and `returnDisplay` (`p_D`) for use in populating items from popups.

Return Value

`Not applicable.`

Parameters

`p_R`
`p_D`

$f_First_field(pNd)

Places the user focus on the a form item (pNd). If pNd is not found then this function places focus on the first found user editable field.

Return Value

```
true (if successful)
```

Parameters

```
pNd
```

GetCookie (pName)

Returns the value of cookie name (pName).

Return Value

Not applicable.

Parameters

pName (String)

SetCookie (pName,pValue)

Sets a cookie (`pName`) to a specified value (`pValue`).

Return Value

`Not applicable.`

Parameters

`pName (String)`
`pValue (String)`

Index

A

C

E

F

J

L

P

R

S

U

V

W