3  APEX_APPLICATION_INSTALL

3.1  Package Overview 3-2
3.2  Attributes Manipulated by APEX_APPLICATION_INSTALL 3-2
3.3  Import Script Examples 3-3
3.4  CLEAR_ALL Procedure 3-5
3.5  GENERATE_APPLICATION_ID Procedure 3-5
3.6  GENERATE_OFFSET Procedure 3-6
3.7  GET_APPLICATION_ALIAS Function 3-7
3.8  GET_APPLICATION_ID Function 3-7
3.9  GET_APPLICATION_NAME Function 3-8
3.10  GET_AUTO_INSTALL_SUP_OBJ Function 3-9
3.11  GET_IMAGE_PREFIX Function 3-9
3.12  GET_KEEP_SESSIONS Function 3-10
3.13  GET_NO_PROXY_DOMAINS Function 3-10
3.14  GET_OFFSET Function 3-11
3.15  GET_PROXY Function 3-12
3.16  GET_REMOTE_SERVER_BASE_URL Function 3-12
3.17  GET_REMOTE_SERVER_HTTPS_HOST Function 3-13
3.18  GET_SCHEMA Function 3-14
3.19  GET_WORKSPACE_ID Function 3-14
3.20  SET_APPLICATION_ALIAS Procedure 3-15
3.21  SET_APPLICATION_ID Procedure 3-16
3.22  SET_APPLICATION_NAME Procedure 3-16
3.23  SET_AUTO_INSTALL_SUP_OBJ Procedure 3-17
3.24  SET_IMAGE_PREFIX Procedure 3-18
3.25  SET_KEEP_SESSIONS Procedure 3-19
3.26  SET_OFFSET Procedure 3-19
3.27  SET_PROXY Procedure 3-20
3.28  SET_REMOTE_SERVER Procedure 3-21
3.29  SET_SCHEMA Procedure 3-22
3.30  SET_WORKSPACE_ID Procedure 3-23
3.31  SET_WORKSPACE Procedure 3-23

4  APEX_AUTHENTICATION

4.1  Constants 4-1
4.2  CALLBACK Procedure 4-1
4.3  CALLBACK 1 Procedure 4-3
4.4  CALLBACK 2 Procedure 4-3
4.5  GET_CALLBACK_URL Function 4-4
4.6  GET_LOGIN_USERNAME_COOKIE Function 4-5
7.22 ADD_MEMBER Procedure
7.23 ADD_MEMBER Function
7.24 ADD_MEMBERS Procedure
7.25 COLLECTION_EXISTS Function
7.26 COLLECTION_HAS_CHANGED Function
7.27 COLLECTION_MEMBER_COUNT Function
7.28 CREATE_COLLECTION Procedure
7.29 CREATE_OR_TRUNCATE_COLLECTION Procedure
7.30 CREATE_COLLECTION_FROM_QUERY Procedure
7.31 CREATE_COLLECTION_FROM_QUERY2 Procedure
7.32 CREATE_COLLECTION_FROM_QUERY_B Procedure
7.33 CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version)
7.34 CREATE_COLLECTION_FROM_QUERYB2 Procedure
7.35 CREATE_COLLECTION_FROM_QUERYB2 Procedure (No bind version)
7.36 DELETE_ALL_COLLECTIONS Procedure
7.37 DELETE_ALL_COLLECTIONS_SESSION Procedure
7.38 DELETE_COLLECTION Procedure
7.39 DELETE_MEMBER Procedure
7.40 DELETE_MEMBERS Procedure
7.41 GET_MEMBER_MD5 Function
7.42 MERGE_MEMBERS Procedure
7.43 MOVE_MEMBER_DOWN Procedure
7.44 MOVE_MEMBER_UP Procedure
7.45 RESEQUENCE_COLLECTION Procedure
7.46 RESET_COLLECTION_CHANGED Procedure
7.47 RESET_COLLECTION_CHANGED_ALL Procedure
7.48 SORT_MEMBERS Procedure
7.49 TRUNCATE_COLLECTION Procedure
7.50 UPDATE_MEMBER Procedure
7.51 UPDATE_MEMBERS Procedure
7.52 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1
7.53 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2
7.54 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3
7.55 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4
7.56 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5
7.57 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6

8 APEX_CREDENTIAL

8.1 CLEAR_TOKENS Procedure
8.2 SET_PERSISTENT_CREDENTIALS Procedure Signature 1
<table>
<thead>
<tr>
<th>Section</th>
<th>Procedure/Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3</td>
<td>SET_PERSISTENT_CREDENTIALS Procedure Signature 2</td>
<td>8-2</td>
</tr>
<tr>
<td>8.4</td>
<td>SET_PERSISTENT_TOKEN Procedure</td>
<td>8-3</td>
</tr>
<tr>
<td>8.5</td>
<td>SET_SESSION_CREDENTIALS Procedure Signature1</td>
<td>8-4</td>
</tr>
<tr>
<td>8.6</td>
<td>SET_SESSION_CREDENTIALS Procedure Signature2</td>
<td>8-5</td>
</tr>
<tr>
<td>8.7</td>
<td>SET_SESSION_TOKEN Procedure</td>
<td>8-5</td>
</tr>
<tr>
<td>9</td>
<td><strong>APEX_CSS</strong></td>
<td></td>
</tr>
<tr>
<td>9.1</td>
<td>ADD Procedure</td>
<td>9-1</td>
</tr>
<tr>
<td>9.2</td>
<td>ADD_3RD_PARTY_LIBRARY_FILE Procedure</td>
<td>9-1</td>
</tr>
<tr>
<td>9.3</td>
<td>ADD_FILE Procedure</td>
<td>9-2</td>
</tr>
<tr>
<td>10</td>
<td><strong>APEX_CUSTOM_AUTH</strong></td>
<td></td>
</tr>
<tr>
<td>10.1</td>
<td>APPLICATION_PAGE_ITEM_EXISTS Function</td>
<td>10-1</td>
</tr>
<tr>
<td>10.2</td>
<td>CURRENT_PAGE_IS_PUBLIC Function</td>
<td>10-2</td>
</tr>
<tr>
<td>10.3</td>
<td>DEFINE_USER_SESSION Procedure</td>
<td>10-3</td>
</tr>
<tr>
<td>10.4</td>
<td>GET_COOKIE_PROPS Procedure</td>
<td>10-3</td>
</tr>
<tr>
<td>10.5</td>
<td>GET_LDAP_PROPS Procedure</td>
<td>10-4</td>
</tr>
<tr>
<td>10.6</td>
<td>GET_NEXT_SESSION_ID Function</td>
<td>10-5</td>
</tr>
<tr>
<td>10.7</td>
<td>GET_SECURITY_GROUP_ID Function</td>
<td>10-6</td>
</tr>
<tr>
<td>10.8</td>
<td>GET_SESSION_ID Function</td>
<td>10-6</td>
</tr>
<tr>
<td>10.9</td>
<td>GET_SESSION_ID_FROM_COOKIE Function</td>
<td>10-7</td>
</tr>
<tr>
<td>10.10</td>
<td>GET_USER Function</td>
<td>10-7</td>
</tr>
<tr>
<td>10.11</td>
<td>GET_USERNAME Function</td>
<td>10-8</td>
</tr>
<tr>
<td>10.12</td>
<td>IS_SESSION_VALID Function</td>
<td>10-8</td>
</tr>
<tr>
<td>10.13</td>
<td>LOGIN Procedure</td>
<td>10-9</td>
</tr>
<tr>
<td>10.14</td>
<td>LOGOUT Procedure [DEPRECATED]</td>
<td>10-10</td>
</tr>
<tr>
<td>10.15</td>
<td>POST_LOGIN Procedure</td>
<td>10-10</td>
</tr>
<tr>
<td>10.16</td>
<td>SESSION_ID_EXISTS Function</td>
<td>10-11</td>
</tr>
<tr>
<td>10.17</td>
<td>SET_SESSION_ID Procedure</td>
<td>10-12</td>
</tr>
<tr>
<td>10.18</td>
<td>SET_SESSION_ID_TO_NEXT_VALUE Procedure</td>
<td>10-12</td>
</tr>
<tr>
<td>10.19</td>
<td>SET_USER Procedure</td>
<td>10-13</td>
</tr>
<tr>
<td>11</td>
<td><strong>APEX_DATA_PARSER</strong></td>
<td></td>
</tr>
<tr>
<td>11.1</td>
<td>Global Constants</td>
<td>11-1</td>
</tr>
<tr>
<td>11.2</td>
<td>Data Types</td>
<td>11-1</td>
</tr>
<tr>
<td>11.3</td>
<td>DISCOVER Function</td>
<td>11-2</td>
</tr>
<tr>
<td>11.4</td>
<td>GET_COLUMNS Function</td>
<td>11-4</td>
</tr>
<tr>
<td>11.5</td>
<td>GET_FILEPROFILE Function</td>
<td>11-5</td>
</tr>
<tr>
<td>11.6</td>
<td>GET_FILETYPE Function</td>
<td>11-8</td>
</tr>
</tbody>
</table>
14  APEX_ERROR

14.1 Constants and Attributes used for Result Types 14-1
14.2 Example of an Error Handling Function 14-3
14.3 ADD_ERROR Procedure Signature 1 14-5
14.4 ADD_ERROR Procedure Signature 2 14-6
14.5 ADD_ERROR Procedure Signature 3 14-7
14.6 ADD_ERROR Procedure Signature 4 14-8
14.7 ADD_ERROR Procedure Signature 5 14-10
14.8 APEX_ERROR.HAVE_ERRORS_OCCURRED Function 14-11
14.9 AUTO_SET_ASSOCIATED_ITEM Procedure 14-12
14.10 EXTRACT_CONSTRAINT_NAME Function 14-12
14.11 GET_FIRST_ORA_ERROR_TEXT Function 14-13
14.12 INIT_ERROR_RESULT Function 14-13

15  APEX_EXEC

15.1 Global Constants 15-3
15.2 Data Types 15-5
15.3 ADD_COLUMN Procedure 15-8
15.4 ADD_DML_ROW Procedure 15-9
15.5 ADD_FILTER Procedure 15-10
15.6 ADD_ORDER_BY Procedure 15-14
15.7 ADD_PARAMETER Procedure 15-15
15.8 CLEAR_DML_ROWS Procedure 15-17
15.9 CLOSE Procedure 15-17
15.10 COPY_DATA Procedure 15-18
15.11 EXECUTE_DML Procedure 15-20
15.12 EXECUTE_PLSQL Procedure 15-20
15.13 EXECUTE_REMOTE_PLSQL Procedure 15-22
15.14 EXECUTE_WEB_SOURCE Procedure 15-23
15.15 GET Functions 15-24
15.16 GET_COLUMN Function 15-26
15.17 GET_COLUMN_COUNT Function 15-26
15.18 GET_COLUMN_POSITION Function 15-26
15.19 GET_DATA_TYPE Functions 15-27
15.20 GET Functions 15-28
15.21 GET_DML_STATUS_CODE Function 15-30
15.22 GET_DML_STATUS_MESSAGE Function 15-31
15.23 GET_PARAMETER Functions 15-31
15.24 GET_ROW_VERSION_CHECKSUM Function 15-33
15.25 GET_TOTAL_ROW_COUNT Function 15-33
15.26 HAS_ERROR Function 15-33
15.27 IS_REMOTE_SQL_AUTH_VALID Function 15-34
15.28 NEXT_ROW Function 15-35
15.29 OPEN_LOCAL_DML_CONTEXT Function 15-35
15.30 OPEN_REMOTE_DML_CONTEXT Function 15-39
15.31 OPEN_REMOTE_SQL_QUERY Function 15-42
15.32 OPEN_QUERY_CONTEXT Procedure 15-44
15.33 OPEN_QUERYCONTEXT Function 15-45
15.34 OPEN_WEB__SOURCE_DML_CONTEXT Function 15-48
15.35 OPEN_WEB_SOURCE_QUERY Function 15-51
15.36 PURGE_WEB_SOURCE_CACHE Procedure 15-53
15.37 SET_NULL Procedure 15-53
15.38 SET_VALUE Procedure 15-54
15.39 SET_VALUES Procedure 15-58
15.40 SET_ROW_VERSION_CHECKSUM Procedure 15-58

16  APEX_EXPORT

16.1 GET_APPLICATION Function 16-1
16.2 GET_WORKSPACE_FILES Function 16-3
16.3 GET_FEEDBACK Function 16-3
16.4 GET_WORKSPACE Function 16-4

17  APEX_INSTANCE_ADMIN

17.1 Available Parameter Values 17-2
17.2 ADD_SCHEMA Procedure 17-9
17.3 ADD_WORKSPACE Procedure 17-9
17.4 CREATE_SCHEMA_EXCEPTION Procedure 17-10
17.5 DB_SIGNATURE Function 17-11
17.6 FREE_WORKSPACE_APP_IDS Procedure 17-12
17.7 GET_PARAMETER Function 17-12
17.8 GET_SCHEMAS Function 17-13
17.9 GET_WORKSPACE_PARAMETER 17-14
17.10 IS_DB_SIGNATURE_VALID Function 17-14
17.11 REMOVE_APPLICATION Procedure 17-15
17.12 REMOVE_SAVED_REPORT Procedure 17-16
17.13 REMOVE_SAVED_REPORTS Procedure 17-16
17.14  REMOVE_SCHEMA Procedure 17-17
17.15  REMOVE_SCHEMA_EXCEPTION Procedure 17-17
17.16  REMOVE_SCHEMA_EXCEPTIONS Procedure 17-18
17.17  REMOVE_SUBSCRIPTION Procedure 17-19
17.18  REMOVE_WORKSPACE Procedure 17-20
17.19  REMOVE_WORKSPACE_EXCEPTIONS Procedure 17-20
17.20  RESERVE_WORKSPACE_APP_IDS Procedure 17-21
17.21  RESTRICT_SCHEMA Procedure 17-23
17.22  SET_LOG_SWITCH_INTERVAL Procedure 17-23
17.23  SET_WORKSPACE_PARAMETER 17-24
17.24  SET_PARAMETER Procedure 17-25
17.25  SET_WORKSPACE_CONSUMER_GROUP Procedure 17-26
17.26  TRUNCATE_LOG Procedure 17-27
17.27  UNRESTRICT_SCHEMA Procedure 17-28

18  APEX_IR

18.1  ADD_FILTER Procedure Signature 1 18-1
18.2  ADD_FILTER Procedure Signature 2 18-3
18.3  CHANGE_SUBSCRIPTION_EMAIL Procedure 18-4
18.4  CHANGE_REPORT_OWNER Procedure 18-5
18.5  CHANGE_SUBSCRIPTION_EMAIL Procedure 18-6
18.6  CHANGE_SUBSCRIPTION_LANG Procedure 18-6
18.7  CLEAR_REPORT Procedure Signature 1 18-7
18.8  CLEAR_REPORT Procedure Signature 2 18-8
18.9  DELETE_REPORT Procedure 18-9
18.10 DELETE_SUBSCRIPTION Procedure 18-10
18.11 GET_LAST VIEWED_REPORT_ID Function 18-10
18.12 GET_REPORT Function 18-11
18.13 RESET_REPORT Procedure Signature 1 18-12
18.14 RESET_REPORT Procedure Signature 2 18-13

19  APEX_ITEM

19.1  CHECKBOX2 Function 19-1
19.2  DATE_POPUP Function 19-3
19.3  DATE_POPUP2 Function 19-5
19.4  DISPLAY_AND_SAVE Function 19-6
19.5  HIDDEN Function 19-7
19.6  MD5_CHECKSUM Function 19-8
19.7  MD5_HIDDEN Function 19-9
19.8   POPUP_FROM_LOV Function 19-10
19.9   POPUP_FROM_QUERY Function 19-12
19.10  POPUPKEY_FROM_LOV Function 19-13
19.11  POPUPKEY_FROM_QUERY Function 19-15
19.12  RADIOGROUP Function 19-16
19.13  SELECT_LIST Function 19-17
19.14  SELECT_LIST_FROM_LOV Function 19-19
19.15  SELECT_LIST_FROM_LOV_XL Function 19-20
19.16  SELECT_LIST_FROM_QUERY Function 19-21
19.17  SELECT_LIST_FROM_QUERY_XL Function 19-23
19.18  SWITCH Function 19-24
19.19  TEXT Function 19-25
19.20  TEXTAREA Function 19-26
19.21  TEXT_FROM_LOV Function 19-27
19.22  TEXT_FROM_LOV_QUERY Function 19-28

20  APEX_JAVASCRIPT

20.1   ADD_3RD_PARTY_LIBRARY_FILE Procedure 20-1
20.2   ADD_ATTRIBUTE Function Signature 1 20-2
20.3   ADD_ATTRIBUTE Function Signature 2 20-3
20.4   ADD_ATTRIBUTE Function Signature 3 20-4
20.5   ADD_ATTRIBUTE Function Signature 4 20-5
20.6   ADDINLINE_CODE Procedure 20-5
20.7   ADD_LIBRARY Procedure 20-6
20.8   ADD_REQUIREJS Procedure 20-7
20.9   ADD_REQUIREJS_DEFINE Procedure 20-8
20.10  ADD_ONLOAD_CODE Procedure 20-8
20.11  ADD_VALUE Function Signature 1 20-9
20.12  ADD_VALUE Function Signature 2 20-10
20.13  ADD_VALUE Function Signature 3 20-10
20.14  ADD_VALUE Function Signature 4 20-11
20.15  Escape Function 20-11

21  APEX_JSON

21.1   Package Overview and Examples 21-2
21.2   Constants and Data Types 21-3
21.3   CLOSE_ALL Procedure 21-4
21.4   CLOSE_ARRAY Procedure 21-5
21.5   CLOSE_OBJECT Procedure 21-5
| 21.6  | DOES_EXIST Function          | 21-5 |
| 21.7  | FIND_PATHS_LIKE Function     | 21-6 |
| 21.8  | FREE_OUTPUT Procedure        | 21-8 |
| 21.9  | FLUSH Procedure              | 21-8 |
| 21.10 | GET_BOOLEAN Function         | 21-9 |
| 21.11 | GET_CLOB_OUTPUT Function     | 21-10|
| 21.12 | GET_COUNT Function           | 21-10|
| 21.13 | GET_DATE Function            | 21-11|
| 21.14 | GET_MEMBERS Function         | 21-12|
| 21.15 | GET_NUMBER Function          | 21-13|
| 21.16 | GET_VALUE Function           | 21-14|
| 21.17 | GET_T_NUMBER Function        | 21-15|
| 21.18 | GET_T_VARCHAR2 Function      | 21-17|
| 21.19 | GET_VARCHAR2 Function        | 21-18|
| 21.20 | GET_CLOB Function            | 21-19|
| 21.21 | INITIALIZE_CLOB_OUTPUT Procedure | 21-20|
| 21.22 | INITIALIZE_OUTPUT Procedure  | 21-21|
| 21.23 | OPEN_ARRAY Procedure         | 21-22|
| 21.24 | OPEN_OBJECT Procedure        | 21-23|
| 21.25 | PARSE Procedure Signature 1  | 21-23|
| 21.26 | PARSE Procedure Signature 2  | 21-24|
| 21.27 | STRINGIFY Function Signature 1 | 21-25|
| 21.28 | STRINGIFY Function Signature 2 | 21-26|
| 21.29 | STRINGIFY Function Signature 3 | 21-26|
| 21.30 | STRINGIFY Function Signature 4 | 21-27|
| 21.31 | TO_MEMBER_NAME Function      | 21-28|
| 21.32 | TO_XMLTYPE Function          | 21-29|
| 21.33 | TO_XMLTYPE_SQL Function      | 21-30|
| 21.34 | WRITE Procedure Signature 1  | 21-30|
| 21.35 | WRITE Procedure Signature 2  | 21-31|
| 21.36 | WRITE Procedure Signature 3  | 21-32|
| 21.37 | WRITE Procedure Signature 4  | 21-32|
| 21.38 | WRITE Procedure Signature 5  | 21-33|
| 21.39 | WRITE Procedure Signature 6  | 21-33|
| 21.40 | WRITE Procedure Signature 7  | 21-34|
| 21.41 | WRITE Procedure Signature 8  | 21-34|
| 21.42 | WRITE Procedure Signature 9  | 21-35|
| 21.43 | WRITE Procedure Signature 10 | 21-36|
| 21.44 | WRITE Procedure Signature 11 | 21-36|
| 21.45 | WRITE Procedure Signature 12 | 21-37|
| 21.46 | WRITE Procedure Signature 13 | 21-37|
22  APEX_JWT

22.1  T_TOKEN 22-1
22.2  ENCODE Function 22-1
22.3  DECODE Function 22-3
22.4  VALIDATE Procedure 22-4

23  APEX_LANG

23.1  CREATE_LANGUAGE_MAPPING Procedure 23-1
23.2  DELETE_LANGUAGE_MAPPING Procedure 23-2
23.3  EMIT_LANGUAGE_SELECTOR_LIST Procedure 23-4
23.4  LANG Function 23-4
23.5  MESSAGE Function 23-5
23.6  PUBLISH_APPLICATION Procedure 23-7
23.7  SEED_TRANSLATIONS Procedure 23-8
23.8  UPDATE_LANGUAGE_MAPPING Procedure 23-9
23.9  UPDATE_MESSAGE Procedure 23-10
23.10  UPDATE_TRANSLATED_STRING Procedure 23-12

24  APEX_LDAP

24.1  AUTHENTICATE Function 24-1
24.2  GET_ALL_USER_ATTRIBUTES Procedure 24-2
24.3  GET_USER_ATTRIBUTES Procedure 24-3
24.4  IS_MEMBER Function 24-5
24.5  MEMBER_OF Function 24-6
24.6  MEMBER_OF2 Function 24-7
24.7  SEARCH Function 24-8

25  APEX_MAIL

25.1  Configuring Oracle Application Express to Send Email 25-2
25.2  ADD_ATTACHMENT Procedure 25-2
25.3  GET_IMAGES_URL Function 25-3
25.4  GET_INSTANCE_URL Function 25-4

ORACLE
25.5  PREPARE_TEMPLATE Procedure  25-5
25.6  PUSH_QUEUE Procedure  25-6
25.7  SEND Procedure  25-7
25.8  SEND Function  25-9
25.9  SEND Procedure  25-10
25.10  SEND Function  25-13

26  APEX_PKG_APP_INSTALL
26.1  DEINSTALL Procedure Signature 1  26-1
26.2  DEINSTALL Procedure Signature 2  26-2
26.3  INSTALL Function Signature 1  26-2
26.4  INSTALL Function Signature 2  26-3
26.5  UPGRADE Procedure Signature 1  26-4
26.6  UPGRADE Procedure Signature 2  26-5

27  APEX_PAGE
27.1  Global Constants  27-1
27.2  IS_DESKTOP_UI Function  27-1
27.3  IS_JQM_SMARTPHONE_UI Function [DEPRECATED]  27-1
27.4  IS_JQM_TABLET_UI Function [DEPRECATED]  27-2
27.5  GET_UI_TYPE Function  27-2
27.6  IS_READ_ONLY Function  27-2
27.7  GET_PAGE_MODE Function  27-2
27.8  PURGE_CACHE Procedure  27-3
27.9  GET_URL Function  27-3

28  APEX_PLUGIN
28.1  Data Types  28-1
28.2  GET_AJAX_IDENTIFIER Function  28-8
28.3  GET_INPUT_NAME_FOR_PAGE_ITEM Function  28-9

29  APEX_PLUGIN_UTIL
29.1  CLEAR_COMPONENT_VALUES Procedure  29-2
29.2  DEBUG_DYNAMIC_ACTION Procedure  29-2
29.3  DEBUG_PAGE_ITEM Procedure Signature 1  29-3
29.4  DEBUG_PAGE_ITEM Procedure Signature 2  29-3
29.5  DEBUG_PROCESS Procedure  29-4
29.6  DEBUG_REGION Procedure Signature 1  29-5
30   APEX_REGION

30.1   IS_READONLY Function  30-1
30.2   PURGE_CACHE Procedure  30-1

31   APEX_SESSION

31.1   SET_DEBUG Procedure  31-1
31.2   SET_TRACE Procedure  31-2
31.3   CREATE_SESSION Procedure  31-3
31.4   DELETE_SESSION Procedure  31-4
31.5   ATTACH Procedure  31-5
31.6   DETACH Procedure  31-6
## APEX_SPATIAL

- **32.1 Data Types**
- **32.2 CHANGE_GEOM_METADATA Procedure**
- **32.3 CIRCLE_POLYGON Function**
- **32.4 DELETE_GEOM_METADATA Procedure**
- **32.5 INSERT_GEOM_METADATA Procedure**
- **32.6 INSERT_GEOM_METADATA_LONLAT Procedure**
- **32.7 POINT Function**
- **32.8 RECTANGLE Function**

## APEX_STRING

- **33.1 FORMAT Function**
- **33.2 GET_INITIALS Function**
- **33.3 GET_SEARCHABLE_PHRASES Function**
- **33.4 GREP Function Signature 1**
- **33.5 GREP Function Signature 2**
- **33.6 GREP Function Signature 3**
- **33.7 JOIN_CLOB Function**
- **33.8 JOIN Function Signature 1**
- **33.9 JOIN Function Signature 2**
- **33.10 NEXT_CHUNK Function**
- **33.11 PLIST_DELETE Procedure**
- **33.12 PLIST_GET Function**
- **33.13 PLIST_PUSH Procedure**
- **33.14 PLIST_PUT Function**
- **33.15 PUSH Procedure Signature 1**
- **33.16 PUSH Procedure Signature 2**
- **33.17 PUSH Procedure Signature 3**
- **33.18 PUSH Procedure Signature 4**
- **33.19 SHUFFLE Function**
- **33.20 SHUFFLE Procedure**
- **33.21 SPLIT Function Signature 1**
- **33.22 SPLIT Function Signature 2**
- **33.23 SPLIT_NUMBERS Function**
- **33.24 STRING_TO_TABLE Function**
- **33.25 TABLE_TO_STRING Function**
### APEX_THEME

<table>
<thead>
<tr>
<th>Section</th>
<th>Function/Procedure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.1</td>
<td>CLEAR_ALL_USERS_STYLE Procedure</td>
<td>34-1</td>
</tr>
<tr>
<td>34.2</td>
<td>CLEAR_USER_STYLE Procedure</td>
<td>34-2</td>
</tr>
<tr>
<td>34.3</td>
<td>DISABLE_USER_STYLE Procedure</td>
<td>34-2</td>
</tr>
<tr>
<td>34.4</td>
<td>ENABLE_USER_STYLE Procedure</td>
<td>34-3</td>
</tr>
<tr>
<td>34.5</td>
<td>GET_USER_STYLE Function</td>
<td>34-4</td>
</tr>
<tr>
<td>34.6</td>
<td>SET_CURRENT_STYLE Procedure</td>
<td>34-5</td>
</tr>
<tr>
<td>34.7</td>
<td>SET_SESSION_STYLE Procedure</td>
<td>34-6</td>
</tr>
<tr>
<td>34.8</td>
<td>SET_SESSION_STYLE_CSS Procedure</td>
<td>34-7</td>
</tr>
<tr>
<td>34.9</td>
<td>SET_USER_STYLE Procedure</td>
<td>34-8</td>
</tr>
</tbody>
</table>

### APEX_UI_DEFAULT_UPDATE

<table>
<thead>
<tr>
<th>Section</th>
<th>Function/Procedure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.1</td>
<td>ADD_AD_COLUMN Procedure</td>
<td>35-2</td>
</tr>
<tr>
<td>35.2</td>
<td>ADD_AD_SYNONYM Procedure</td>
<td>35-3</td>
</tr>
<tr>
<td>35.3</td>
<td>DEL_AD_COLUMN Procedure</td>
<td>35-4</td>
</tr>
<tr>
<td>35.4</td>
<td>DEL_AD_SYNONYM Procedure</td>
<td>35-5</td>
</tr>
<tr>
<td>35.5</td>
<td>DEL_COLUMN Procedure</td>
<td>35-5</td>
</tr>
<tr>
<td>35.6</td>
<td>DEL_GROUP Procedure</td>
<td>35-6</td>
</tr>
<tr>
<td>35.7</td>
<td>DEL_TABLE Procedure</td>
<td>35-7</td>
</tr>
<tr>
<td>35.8</td>
<td>SYNCH_TABLE Procedure</td>
<td>35-7</td>
</tr>
<tr>
<td>35.9</td>
<td>UPD_AD_COLUMN Procedure</td>
<td>35-8</td>
</tr>
<tr>
<td>35.10</td>
<td>UPD_AD_SYNONYM Procedure</td>
<td>35-9</td>
</tr>
<tr>
<td>35.11</td>
<td>UPD_COLUMN Procedure</td>
<td>35-10</td>
</tr>
<tr>
<td>35.12</td>
<td>UPDDISPLAY_IN_FORM Procedure</td>
<td>35-12</td>
</tr>
<tr>
<td>35.13</td>
<td>UPDDISPLAY_IN_REPORT Procedure</td>
<td>35-13</td>
</tr>
<tr>
<td>35.14</td>
<td>UPD_FORM_REGION_TITLE Procedure</td>
<td>35-13</td>
</tr>
<tr>
<td>35.15</td>
<td>UPD_GROUP Procedure</td>
<td>35-14</td>
</tr>
<tr>
<td>35.16</td>
<td>UPD_ITEM_DISPLAY_HEIGHT Procedure</td>
<td>35-15</td>
</tr>
<tr>
<td>35.17</td>
<td>UPD_ITEM_DISPLAY_WIDTH Procedure</td>
<td>35-16</td>
</tr>
<tr>
<td>35.18</td>
<td>UPD_ITEM_FORMAT_MASK Procedure</td>
<td>35-16</td>
</tr>
<tr>
<td>35.19</td>
<td>UPD_ITEM_HELP Procedure</td>
<td>35-17</td>
</tr>
<tr>
<td>35.20</td>
<td>UPD_LABEL Procedure</td>
<td>35-18</td>
</tr>
<tr>
<td>35.21</td>
<td>UPD_REPORT_ALIGNMENT Procedure</td>
<td>35-18</td>
</tr>
<tr>
<td>35.22</td>
<td>UPD_REPORT_FORMAT_MASK Procedure</td>
<td>35-19</td>
</tr>
<tr>
<td>35.23</td>
<td>UPD_REPORT_REGION_TITLE Procedure</td>
<td>35-20</td>
</tr>
<tr>
<td>35.24</td>
<td>UPD_TABLE Procedure</td>
<td>35-20</td>
</tr>
<tr>
<td>Section</td>
<td>Function/Procedure</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------</td>
<td>------</td>
</tr>
<tr>
<td>36.1</td>
<td>CACHE_GET_DATE_OF_PAGE_CACHE Function</td>
<td>36-5</td>
</tr>
<tr>
<td>36.2</td>
<td>CACHE_GET_DATE_OF_REGION_CACHE Function</td>
<td>36-6</td>
</tr>
<tr>
<td>36.3</td>
<td>CACHE_PURGE_BY_APPLICATION Procedure</td>
<td>36-6</td>
</tr>
<tr>
<td>36.4</td>
<td>CACHE_PURGE_BY_PAGE Procedure</td>
<td>36-7</td>
</tr>
<tr>
<td>36.5</td>
<td>CACHE_PURGE_STALE Procedure</td>
<td>36-8</td>
</tr>
<tr>
<td>36.6</td>
<td>CHANGE_CURRENT_USER_PW Procedure</td>
<td>36-8</td>
</tr>
<tr>
<td>36.7</td>
<td>CHANGE_PASSWORD_ON_FIRST_USE Function</td>
<td>36-9</td>
</tr>
<tr>
<td>36.8</td>
<td>CLOSE_OPEN_DB_LINKS Procedure</td>
<td>36-10</td>
</tr>
<tr>
<td>36.9</td>
<td>CLEAR_APP_CACHE Procedure</td>
<td>36-11</td>
</tr>
<tr>
<td>36.10</td>
<td>CLEAR_PAGE_CACHE Procedure</td>
<td>36-11</td>
</tr>
<tr>
<td>36.11</td>
<td>CLEAR_USER_CACHE Procedure</td>
<td>36-12</td>
</tr>
<tr>
<td>36.12</td>
<td>COUNT_CLICK Procedure</td>
<td>36-12</td>
</tr>
<tr>
<td>36.13</td>
<td>CREATE_USER Procedure</td>
<td>36-13</td>
</tr>
<tr>
<td>36.14</td>
<td>CREATE_USER_GROUP Procedure</td>
<td>36-17</td>
</tr>
<tr>
<td>36.15</td>
<td>CURRENT_USER_IN_GROUP Function</td>
<td>36-18</td>
</tr>
<tr>
<td>36.16</td>
<td>CUSTOM_CALENDAR Procedure</td>
<td>36-18</td>
</tr>
<tr>
<td>36.17</td>
<td>DELETE_USER_GROUP Procedure Signature 1</td>
<td>36-19</td>
</tr>
<tr>
<td>36.18</td>
<td>DELETE_USER_GROUP Procedure Signature 2</td>
<td>36-20</td>
</tr>
<tr>
<td>36.19</td>
<td>DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1</td>
<td>36-20</td>
</tr>
<tr>
<td>36.20</td>
<td>DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2</td>
<td>36-21</td>
</tr>
<tr>
<td>36.21</td>
<td>DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3</td>
<td>36-23</td>
</tr>
<tr>
<td>36.22</td>
<td>DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4</td>
<td>36-24</td>
</tr>
<tr>
<td>36.23</td>
<td>EDIT_USER Procedure</td>
<td>36-25</td>
</tr>
<tr>
<td>36.24</td>
<td>END_USER_ACCOUNT_DAYS_LEFT Function</td>
<td>36-30</td>
</tr>
<tr>
<td>36.25</td>
<td>EXPIRE_END_USER_ACCOUNT Procedure</td>
<td>36-30</td>
</tr>
<tr>
<td>36.26</td>
<td>EXPIRE_WORKSPACE_ACCOUNT Procedure</td>
<td>36-31</td>
</tr>
<tr>
<td>36.27</td>
<td>EXPORT_USERS Procedure</td>
<td>36-32</td>
</tr>
<tr>
<td>36.28</td>
<td>FETCH_APP_ITEM Function</td>
<td>36-33</td>
</tr>
<tr>
<td>36.29</td>
<td>FETCH_USER Procedure Signature 1</td>
<td>36-34</td>
</tr>
<tr>
<td>36.30</td>
<td>FETCH_USER Procedure Signature 2</td>
<td>36-36</td>
</tr>
<tr>
<td>36.31</td>
<td>FETCH_USER Procedure Signature 3</td>
<td>36-38</td>
</tr>
<tr>
<td>36.32</td>
<td>FIND_SECURITY_GROUP_ID Function</td>
<td>36-41</td>
</tr>
<tr>
<td>36.33</td>
<td>FIND_WORKSPACE Function</td>
<td>36-42</td>
</tr>
<tr>
<td>36.34</td>
<td>GET_ACCOUNT_LOCKED_STATUS Function</td>
<td>36-43</td>
</tr>
<tr>
<td>36.35</td>
<td>GET_APPLICATION_STATUS Function</td>
<td>36-43</td>
</tr>
<tr>
<td>36.36</td>
<td>GET_ATTRIBUTE Function</td>
<td>36-44</td>
</tr>
<tr>
<td>36.37</td>
<td>GET_AUTHENTICATION_RESULT Function</td>
<td>36-45</td>
</tr>
<tr>
<td>36.38</td>
<td>GET_BLOB_FILE_SRC Function</td>
<td>36-46</td>
</tr>
<tr>
<td>36.39</td>
<td>GET_BUILD_OPTION_STATUS Function Signature 1</td>
<td>36-47</td>
</tr>
</tbody>
</table>
36.40 GET_BUILD_OPTION_STATUS Function Signature 2 36-48
36.41 GET_CURRENT_USER_ID Function 36-49
36.42 GET_DEFAULT_SCHEMA Function 36-49
36.43 GET_EDITION Function 36-50
36.44 GET_EMAIL Function 36-50
36.45 GET_FEEDBACK_FOLLOW_UP Function 36-51
36.46 GET_FILE Procedure 36-52
36.47 GET_FILE_ID Function 36-53
36.48 GET_FIRST_NAME Function 36-54
36.49 GET_GLOBAL_NOTIFICATION Function 36-54
36.50 GET_GROUPS_USER_BELONGS_TO Function 36-55
36.51 GET_GROUP_ID Function 36-56
36.52 GET_GROUP_NAME Function 36-56
36.53 GET_HASH Function 36-57
36.54 GET_HIGH_CONTRAST_MODE_TOGGLE Function 36-58
36.55 GET_LAST_NAME Function 36-59
36.56 GET_NUMERIC_SESSION_STATE Function 36-60
36.57 GET_PREFERENCE Function 36-61
36.58 GET_PRINT_DOCUMENT Function Signature 1 36-62
36.59 GET_PRINT_DOCUMENT Function Signature 2 36-62
36.60 GET_PRINT_DOCUMENT Function Signature 3 36-63
36.61 GET_PRINT_DOCUMENT Function Signature 4 36-64
36.62 GET_SCREEN_READER_MODE_TOGGLE Function 36-65
36.63 GET_SESSION_LANG Function 36-66
36.64 GET_SESSION_STATE Function 36-66
36.65 GET_SESSION_TERRITORY Function 36-67
36.66 GET_SESSION_TIME_ZONE Function 36-68
36.67 GET_SINCE Function 36-68
36.68 GET_SUPPORTING_OBJECT_SCRIPT Function 36-70
36.69 GET_SUPPORTING_OBJECT_SCRIPT Procedure 36-71
36.70 GET_USER_ID Function 36-72
36.71 GET_USER_ROLES Function 36-72
36.72 GET_USERNAME Function 36-73
36.73 HOST_URL Function 36-74
36.74 HTML_PCT_GRAPH_MASK Function 36-75
36.75 INCREMENT_CALENDAR Procedure 36-76
36.76 IR_CLEAR Procedure [DEPRECATED] 36-76
36.77 IR_DELETE_REPORT Procedure [DEPRECATED] 36-77
36.78 IR_DELETE_SUBSCRIPTION Procedure [DEPRECATED] 36-78
36.79 IR_FILTER Procedure [DEPRECATED] 36-79
36.80 IR_RESET Procedure [DEPRECATED] 36-80
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.81</td>
<td>IS_HIGH_CONTRAST_SESSION Function</td>
</tr>
<tr>
<td>36.82</td>
<td>IS_HIGH_CONTRAST_SESSION_YN Function</td>
</tr>
<tr>
<td>36.83</td>
<td>IS_LOGIN_PASSWORD_VALID Function</td>
</tr>
<tr>
<td>36.84</td>
<td>IS_SCREEN_READER_SESSION Function</td>
</tr>
<tr>
<td>36.85</td>
<td>IS_SCREEN_READER_SESSION_YN Function</td>
</tr>
<tr>
<td>36.86</td>
<td>IS_USERNAME_UNIQUE Function</td>
</tr>
<tr>
<td>36.87</td>
<td>KEYVAL_NUM Function</td>
</tr>
<tr>
<td>36.88</td>
<td>KEYVAL_VC2 Function</td>
</tr>
<tr>
<td>36.89</td>
<td>LOCK_ACCOUNT Procedure</td>
</tr>
<tr>
<td>36.90</td>
<td>PASSWORD_FIRST_USE_OCCURRED Function</td>
</tr>
<tr>
<td>36.91</td>
<td>PREPARE_URL Function</td>
</tr>
<tr>
<td>36.92</td>
<td>PUBLIC_CHECK_AUTHORIZATION Function [DEPRECATED]</td>
</tr>
<tr>
<td>36.93</td>
<td>PURGE_REGIONS_BY_APP Procedure</td>
</tr>
<tr>
<td>36.94</td>
<td>PURGE_REGIONS_BY_NAME Procedure</td>
</tr>
<tr>
<td>36.95</td>
<td>PURGE_REGIONS_BY_PAGE Procedure</td>
</tr>
<tr>
<td>36.96</td>
<td>REDIRECT_URL Procedure</td>
</tr>
<tr>
<td>36.97</td>
<td>REMOVE_PREFERENCE Procedure</td>
</tr>
<tr>
<td>36.98</td>
<td>REMOVE_SORT_PREFERENCES Procedure</td>
</tr>
<tr>
<td>36.99</td>
<td>REMOVE_USER Procedure</td>
</tr>
<tr>
<td>36.100</td>
<td>RESET_AUTHORIZATIONS Procedure [DEPRECATED]</td>
</tr>
<tr>
<td>36.101</td>
<td>RESET_PASSWORD Procedure</td>
</tr>
<tr>
<td>36.102</td>
<td>RESET_PW Procedure</td>
</tr>
<tr>
<td>36.103</td>
<td>SAVEKEY_NUM Function</td>
</tr>
<tr>
<td>36.104</td>
<td>SAVEKEY_VC2 Function</td>
</tr>
<tr>
<td>36.105</td>
<td>SET_APP_BUILD_STATUS Procedure</td>
</tr>
<tr>
<td>36.106</td>
<td>SET_APPLICATION_STATUS Procedure</td>
</tr>
<tr>
<td>36.107</td>
<td>SET_ATTRIBUTE Procedure</td>
</tr>
<tr>
<td>36.108</td>
<td>SET_AUTHENTICATION_RESULT Procedure</td>
</tr>
<tr>
<td>36.109</td>
<td>SET_BUILD_OPTION_STATUS Procedure</td>
</tr>
<tr>
<td>36.110</td>
<td>SET_CURRENT_THEME_STYLE Procedure [DEPRECATED]</td>
</tr>
<tr>
<td>36.111</td>
<td>SET_CUSTOM_AUTH_STATUS Procedure</td>
</tr>
<tr>
<td>36.112</td>
<td>SET_EDITION Procedure</td>
</tr>
<tr>
<td>36.113</td>
<td>SET_EMAIL Procedure</td>
</tr>
<tr>
<td>36.114</td>
<td>SET_FIRST_NAME Procedure</td>
</tr>
<tr>
<td>36.115</td>
<td>SET_GLOBAL_NOTIFICATION Procedure</td>
</tr>
<tr>
<td>36.116</td>
<td>SET_GROUP_GROUP_GRANTS Procedure</td>
</tr>
<tr>
<td>36.117</td>
<td>SET_GROUP_USER_GRANTS Procedure</td>
</tr>
<tr>
<td>36.118</td>
<td>SET_LAST_NAME Procedure</td>
</tr>
<tr>
<td>36.119</td>
<td>SET_PARSING_SCHEMA_FOR_REQUEST Procedure</td>
</tr>
<tr>
<td>36.120</td>
<td>SET_PREFERENCE Procedure</td>
</tr>
<tr>
<td>36.121</td>
<td>SET_SECURITY_GROUP_ID Procedure</td>
</tr>
</tbody>
</table>
36.122  SET_SESSION_HIGH_CONTRAST_OFF Procedure  36-116
36.123  SET_SESSION_HIGH_CONTRAST_ON Procedure  36-116
36.124  SET_SESSION_LANG Procedure  36-117
36.125  SET_SESSION_LIFETIME_SECONDS Procedure  36-117
36.126  SET_SESSION_MAX_IDLE_SECONDS Procedure  36-118
36.127  SET_SESSION_SCREEN_READER_OFF Procedure  36-119
36.128  SET_SESSION_SCREEN_READER_ON Procedure  36-120
36.129  SET_SESSION_STATE Procedure  36-120
36.130  SET_SESSION_TERRITORY Procedure  36-121
36.131  SET_SESSION_TIME_ZONE Procedure  36-122
36.132  SET_USERNAME Procedure  36-122
36.133  SET_WORKSPACE Procedure  36-123
36.134  SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure  36-124
36.135  SHOW_SCREEN_READER_MODE_TOGGLE Procedure  36-125
36.136  STRING_TO_TABLE Function [DEPRECATED]  36-126
36.137  STRONG_PASSWORD_CHECK Procedure  36-127
36.138  STRONG_PASSWORD_VALIDATION Function  36-130
36.139  SUBMIT_FEEDBACK Procedure  36-131
36.140  SUBMIT_FEEDBACK_FOLLOWUP Procedure  36-133
36.141  TABLE_TO_STRING Function [DEPRECATED]  36-134
36.142  UNEXPIRE_END_USER_ACCOUNT Procedure  36-135
36.143  UNEXPIRE_WORKSPACE_ACCOUNT Procedure  36-136
36.144  UNLOCK_ACCOUNT Procedure  36-137
36.145  URL_ENCODE Function  36-138
36.146  WORKSPACE_ACCOUNT_DAYS_LEFT Function  36-139

37  APEX_WEB_SERVICE

37.1  About the APEX_WEB_SERVICE API  37-1
37.2  Invoking a SOAP Style Web Service  37-2
37.3  Invoking a RESTful Style Web Service  37-3
37.4  Retrieving Cookies and HTTP Headers  37-4
37.5  Setting Cookies and HTTP Headers  37-5
37.6  BLOB2CLOBBASE64 Function  37-5
37.7  CLOBBASE642BLOB Function  37-6
37.8  MAKE_REQUEST Procedure  37-7
37.9  MAKE_REQUEST Function  37-9
37.10  MAKE_REST_REQUEST Procedure  37-10
37.11  MAKE_REST_REQUEST_B Function  37-12
37.12  OAUTH_AUTHENTICATE Function  37-14
37.13  OAUTH_GET_LAST_TOKEN Function  37-15
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>37.14</td>
<td>OAUTH_SET_TOKEN Function</td>
<td>37-15</td>
</tr>
<tr>
<td>37.15</td>
<td>PARSE_RESPONSE Function</td>
<td>37-16</td>
</tr>
<tr>
<td>37.16</td>
<td>PARSE_RESPONSE_CLOB Function</td>
<td>37-17</td>
</tr>
<tr>
<td>37.17</td>
<td>PARSE_XML Function</td>
<td>37-17</td>
</tr>
<tr>
<td>37.18</td>
<td>PARSE_XML_CLOB Function</td>
<td>37-19</td>
</tr>
<tr>
<td>38</td>
<td>APEX_ZIP</td>
<td></td>
</tr>
<tr>
<td>38.1</td>
<td>Data Types</td>
<td>38-1</td>
</tr>
<tr>
<td>38.2</td>
<td>ADD_FILE Procedure</td>
<td>38-1</td>
</tr>
<tr>
<td>38.3</td>
<td>FINISH Procedure</td>
<td>38-2</td>
</tr>
<tr>
<td>38.4</td>
<td>GET_FILE_CONTENT Function</td>
<td>38-3</td>
</tr>
<tr>
<td>38.5</td>
<td>GET_FILES Function</td>
<td>38-3</td>
</tr>
<tr>
<td>39</td>
<td>JavaScript APIs</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Using REST Administration Interface API</td>
<td></td>
</tr>
<tr>
<td>40.1</td>
<td>Authentication</td>
<td>40-1</td>
</tr>
<tr>
<td>40.2</td>
<td>Individual REST Services</td>
<td>40-3</td>
</tr>
<tr>
<td>40.2.1</td>
<td>Fetch Instance-Level Statistics</td>
<td>40-3</td>
</tr>
<tr>
<td>40.2.2</td>
<td>Fetch Workspace-Level Statistics</td>
<td>40-6</td>
</tr>
<tr>
<td>40.2.3</td>
<td>Application-Level Statistics</td>
<td>40-9</td>
</tr>
<tr>
<td>40.2.4</td>
<td>Instance Overview</td>
<td>40-11</td>
</tr>
<tr>
<td>40.2.5</td>
<td>REST Service Version Information</td>
<td>40-13</td>
</tr>
</tbody>
</table>

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

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

**Audience**

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

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

**See Also:**

Oracle Application Express App Builder User's Guide

**Documentation Accessibility**


**Access to Oracle Support**

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

For more information, see these Oracle resources:

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

Conventions

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

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

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><code>monospace</code></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Changes in This Release

This preface contains:

• Changes in Oracle Application Express Release 19.1

Changes in Oracle Application Express Release 19.1

The following are changes in Oracle Application Express API Reference for Oracle Application Express release 19.1.

• New Features
• Deprecated and Desupported Features

New Features

The following features are new in this release:

• APEX_DATA_PARSER (New)
  This is a new package contains the implementation for the file parser in APEX. APEX_DATA_PARSER supports XML, JSON, CSV and XLSX files.
  DISCOVER Function - This is a function to discover the column profile of a file.
  GET_COLUMNS Function - This function returns the columns of a parser profile as a table in order to be consumed by APEX components.
  GET_FILE_PROFILE Function - This function returns the current file profile in JSON format.
  GET_FILE_TYPE Function - This function returns a file type, based on a file name extension.
  GET_XLSX_WORKSHEETS Function - This function returns information on worksheets within an XLSX workbook as a list of apex_t_parser_worksheet instances.
  JSON_TO_PROFILE Function - This function converts a file profile in JSON format to an instance of the t_file_profile record type.
  PARSE Function - This is the main parser function. It allows to parse XML, XLSX, CSV or JSON files and returns a generic table

• APEX_EXEC (Updates)
  Added functions and procedures
  GET_DATA_TYPE Functions - This function converts the t_data_type constant into the VARCHAR2 representation. Converts a data type VARCHAR2 representation to the t_data_type constant.
  ADD_DML_ROW Procedure - This procedure adds one row to the DML context.
CLEAR_DML_ROWS Procedure - This procedure clears all DML rows which have been added with add_dml_rows.

COPY_DATA Procedure - This procedure fetches all rows from the source context and writes to the target context.

EXECUTE_DML Procedure - This procedure executes the DML context.

GET_FUNCTIONS - This function retrieves column values for different data types.

GET_DML_STATUS_CODE Function - This function returns the SQL status code of the last context execution, for the current row.

GET_DML_STATUS_MESSAGE Function - This function returns the SQL status message of the last context execution, for the current row.

GET_ROW_VERSION_CHECKSUM Function - This function returns the row version checksum for the current row.

HAS_ERROR Function - This function returns the when DML execution led to an error and false, when not.

OPEN_LOCAL_DML_CONTEXT Function - This function opens a DML context based for a local database.

OPEN_REMOTE_DML_CONTEXT Function - This function opens a DML context based for a remote database.

OPEN_WEB__SOURCE_DML_CONTEXT Function - This function opens a DML context based for a web source module.

SET_NULL Procedure - This procedure sets procedures to set a DML column value to NULL.

SET_VALUE Procedure - This procedure sets DML column values for different data types.

SET_VALUES Procedure - This procedure sets all column values in the DML context with corresponding column values from the source (query) context.

SET_ROW_VERSION_CHECKSUM Procedure - This procedure sets the row version checksum to use for lost update detection for the current DML row.

• APEX_SESSION (Updates)
  Added a new procedure.

CREATE_SESSION Procedure - This procedure creates a new session for the given application, set environment and run the application's Initialization PL/SQL Code.

• APEX_INSTANCE_ADMIN (Updates)
  Added new functions.

DB_SIGNATURE Function - This function computes the current database signature value.

IS_DB_SIGNATURE_VALID - This function returns whether the instance parameter DB_SIGNATURE matches the value of the function db_signature.

Available Parameter Values (Updates) - Added a new parameter DB_SIGNATURE, INSTANCE_NO_PROXY_DOMAINS, HEADER_AUTH_CALLBACK, and SOCIAL_AUTH_CALLBACK.

• APEX_STRING (Updates)
Added procedures and functions

PLIST_PUSH procedure - This procedure appends key/value to the property list, without looking for duplicates.

PUSH Procedure Signature 4 - This procedure appends values of a PL/SQL table to apex_t_vvarchar2 table.

STRING_TO_TABLE Function - This function returns the split input at separator, returning a vc_arr2.

TABLE_TO_STRING Function - This function returns the values of the apex_application_global.vc_arr2 input table p_table as a concatenated varchar2, separated by p_sep.

GET_SEARCHABLE_PHRASES Function - This function returns distinct phrases of 1-3 consecutive lower case words in the input strings.

Updated PLIST_DELETE, PUSH Signature1, and PUSH Signature 2.

• APEX_AUTHENTICATION (Updates)
  Added a procedure
  CALLBACK 1 Procedure - This procedure is the landing resource for OAuth2-based authentication schemes.

• APEX_UTIL (Updates)
  Added a function
  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.

Deprecated and Desupported Features

See Deprecated Features and Desupported Features in Oracle Application Express Release Notes.
The `APEX_APPLICATION` package is a PL/SQL package that implements the Oracle Application Express rendering engine. You can use this package to take advantage of many global variables. “Global Variables Available in APEX_APPLICATION” describes the global variables available in the `APEX_APPLICATION` package.

### Note:
"Global Variables"

- Global Variables
- Referencing Arrays
- Referencing Values Within an On Submit Process
- Converting an Array to a Single Value
- HELP Procedure
- STOP_APEX_ENGINE Procedure

## 1.1 Global Variables

### Table 1-1  Global Variables Available in APEX_APPLICATION

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_USER</td>
<td>Specifies the currently logged in user.</td>
</tr>
<tr>
<td>G_FLOW_ID</td>
<td>Specifies the ID of the currently running application.</td>
</tr>
<tr>
<td>G_FLOW_STEP_ID</td>
<td>Specifies the ID of the currently running page.</td>
</tr>
<tr>
<td>G_FLOW_OWNER</td>
<td>Defaults to the application's parsing schema. Use #OWNER# to reference this value in SQL queries and PL/SQL.</td>
</tr>
<tr>
<td>G_REQUEST</td>
<td>Specifies the value of the request variable most recently passed to or set within the show or accept modules.</td>
</tr>
<tr>
<td>G_BROWSER_LANGUAGE</td>
<td>Refers to the web browser's current language preference.</td>
</tr>
</tbody>
</table>

### Note:
Changing `G_FLOW_OWNER` at runtime does not change the parsing schema.
Table 1-1 (Cont.) Global Variables Available in APEX_APPLICATION

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

1.2 Referencing Arrays

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

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

```html
<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>
</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:

```plaintext
<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.

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

### 1.3 Referencing Values Within an On Submit Process

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

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

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

### 1.4 Converting an Array to a Single Value

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

```plaintext
htp.p(APEX_UTIL.TABLE_TO_STRING(APEX_APPLICATION.G_F01));
```
This function enables you to reference `G_F01` to `G_F50` values in an application process that performs actions on data. The following sample process demonstrates how values are inserted into a table:

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

### 1.5 HELP Procedure

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

**Syntax**

```
APEX_APPLICATION.HELP ( 
  p_request        IN VARCHAR2 DEFAULT NULL, 
  p_flow_id        IN VARCHAR2 DEFAULT NULL, 
  p_flow_step_id   IN VARCHAR2 DEFAULT NULL, 
  p_show_item_help IN VARCHAR2 DEFAULT 'YES', 
  p_show_regions  IN VARCHAR2 DEFAULT 'YES', 
  p_before_page_html IN VARCHAR2 DEFAULT '<p>', 
  p_after_page_html IN VARCHAR2 DEFAULT NULL, 
  p_before_region_html IN VARCHAR2 DEFAULT NULL, 
  p_after_region_html IN VARCHAR2 DEFAULT '</td></tr></table></p>', 
  p_before_prompt_html IN VARCHAR2 DEFAULT '<p><b>', 
  p_after_prompt_html IN VARCHAR2 DEFAULT '</b></p>:&nbsp;', 
  p_before_item_html IN VARCHAR2 DEFAULT NULL, 
  p_after_item_html IN VARCHAR2 DEFAULT NULL); 
```

**Parameters**

*Table 1-2* describes the parameters available in the HELP procedure.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_request</code></td>
<td>Not used.</td>
</tr>
<tr>
<td><code>p_flow_id</code></td>
<td>The application ID that contains the page or item level help you want to output.</td>
</tr>
<tr>
<td><code>p_flow_step_id</code></td>
<td>The page ID that contains the page or item level help you want to display.</td>
</tr>
<tr>
<td><code>p_show_item_help</code></td>
<td>Flag to determine if item level help is output. If this parameter is supplied, the value must be either 'YES' or 'NO', if not the default value is 'YES'.</td>
</tr>
<tr>
<td><code>p_show_regions</code></td>
<td>Flag to determine if region headers are output (for regions containing page items). If this parameter is supplied, the value must be either 'YES' or 'NO', if not the default value is 'YES'.</td>
</tr>
<tr>
<td><code>p_before_page_html</code></td>
<td>Use this parameter to include HTML between the page level help text and item level help text.</td>
</tr>
</tbody>
</table>
### Table 1-2 (Cont.) HELP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_after_page_html</td>
<td>Use this parameter to include HTML at the bottom of the output, after all other help.</td>
</tr>
<tr>
<td>p_before_region_html</td>
<td>Use this parameter to include HTML before every region section. Note this parameter is ignored if p_show_regions is set to 'NO'.</td>
</tr>
<tr>
<td>p_after_region_html</td>
<td>Use this parameter to include HTML after every region section. Note this parameter is ignored if p_show_regions is set to 'NO'.</td>
</tr>
<tr>
<td>p_before_prompt_html</td>
<td>Use this parameter to include HTML before every item label for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.</td>
</tr>
<tr>
<td>p_after_prompt_html</td>
<td>Use this parameter to include HTML after every item label for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.</td>
</tr>
<tr>
<td>p_before_item_html</td>
<td>Use this parameter to include HTML before every item help text for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.</td>
</tr>
<tr>
<td>p_after_item_html</td>
<td>Use this parameter to include HTML after every item help text for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.</td>
</tr>
</tbody>
</table>

### Example

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

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

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

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

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

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

## 1.6 STOP_APEX_ENGINE Procedure

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

### Note:

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

**Syntax**

```
APEX_APPLICATION.STOP_APEX_ENGINE
```

**Parameters**

None

**Example 1**

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

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

**Example 2**

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

```
begin
  ... code which can raise an exception ...
  owa_util.redirect_url('http://apex.oracle.com');
apex_application.stop_apex_engine;
exception
  when apex_application.e_stop_apex_engine then
    raise; -- raise again the stop Application Express engine exception
  when others then
    ...; -- code to handle the exception
end;
```
The APEX_ACL package provides utilities that you can use when programming in the Oracle Application Express environment related to application access control shared components. You can use APEX_ACL package to add, remove, or replace user roles. You can also take advantage of INSTEAD OF trigger on APEX_APPL_ACL_USERS view to edit user roles with DML statements (INSERT, UPDATE, and DELETE). If the package is used outside of Oracle APEX environment, the security_group_id must be set using either APEX_UTIL.SET_WORKSPACE or APEX_UTIL.SET_SECURITY_GROUP_ID before the call. The related APEX views to get more information on application users and roles are APEX_APPL_ACL_ROLES, APEX_APPL_ACL_USER_ROLES, and APEX_APPL_ACL_USERS.

- ADD_USER_ROLE Procedure Signature 1
- ADD_USER_ROLE Procedure Signature 2
- HAS_USER_ANY_ROLES Function
- HAS_USER_ROLE Function
- REMOVE_USER_ROLE Procedure Signature 1
- REMOVE_USER_ROLE Procedure Signature 2
- REPLACE_USER_ROLES Procedure Signature 1
- REPLACE_USER_ROLES Procedure Signature 2
- REMOVE_ALL_USER_ROLES Procedure

## 2.1 ADD_USER_ROLE Procedure Signature 1

This procedure assigns a role to a user.

### Syntax

APEX_ACL.ADD_USER_ROLE (  
  p_application_id in number default wwv_flow_security.g_flow_id,  
  p_user_name      in varchar2,  
  p_role_id        in number );

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to assign role to a user. Defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to assign the role to.</td>
</tr>
</tbody>
</table>
Table 2-1 (Cont.) ADD_USER_ROLE Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_role_id</td>
<td>The ID of the role.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use ADD_USER_ROLE procedure to assign role ID of 2505704029884282 to the user name called 'SCOTT' in application 255.

```sql
begin
APEX_ACL.ADD_USER_ROLE (
    p_application_id => 255,
    p_user_name      => 'SCOTT',
    p_role_id        => 2505704029884282);
end;
```

2.2 ADD_USER_ROLE Procedure Signature 2

This procedure assigns a role to a user.

Syntax

```sql
APEX_ACL.ADD_USER_ROLE (
    p_application_id in number   default wwv_flow_security.g_flow_id,
    p_user_name      in varchar2,
    p_role_static_id in varchar2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to assign role to a user. Defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to assign the role to.</td>
</tr>
<tr>
<td>p_role_static_id</td>
<td>The case insensitive name of the role static ID.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use ADD_USER_ROLE procedure to assign role static ID 'ADMINISTRATOR' to the user name called 'SCOTT' in application 255.

```sql
begin
APEX_ACL.ADD_USER_ROLE (
    p_application_id => 255,
    p_user_name      => 'SCOTT',
```


2.3 HAS_USER_ANY_ROLES Function

This function returns `TRUE` if, the user is assigned to any application role. This function can be used to check if a user is allowed to access an application.

Syntax

```sql
APEX_ACL.HAS_USER_ANY_ROLES (    p_application_id in number   default wwv_flow_security.g_flow_id,    p_user_name    in varchar2 default wwv_flow.g_user )    return boolean;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to check if a user is assigned to any application role. It defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to check. Defaults to the current logged in user.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use `HAS_USER_ANY_ROLES` function to check if the user name called `SCOTT` is assigned to any application role in application 255.

```sql
begin    return APEX_ACL.HAS_USER_ANY_ROLES (    p_application_id => 255,    p_user_name => 'SCOTT' );
end;
```

2.4 HAS_USER_ROLE Function

This function returns `TRUE` if, the user is assigned to the specified role.

Syntax

```sql
APEX_ACL.HAS_USER_ROLE (    p_application_id in number   default wwv_flow_security.g_flow_id,    p_user_name    in varchar2 default wwv_flow.g_user,    p_role_static_id in varchar2 )    return boolean;
```
Parameters

Table 2-4  HAS_USER_ROLE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to check if a user is assigned to the specific role. Defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to check. It defaults to the current logged in user.</td>
</tr>
<tr>
<td>p_role_static_id</td>
<td>The case insensitive name of the role static ID.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use HAS_USER_ROLE function to check if the user name called 'SCOTT' is assigned to role static IDs of 'ADMINISTRATOR' in application 255.

declare
  l_is_admin boolean := false;
bEGIN
  l_is_admin := APEX_ACL.HAS_USER_ROLE ( p_application_id => 255, p_user_name => 'SCOTT', p_role_static_id => 'ADMINISTRATOR' );
  if not l_is_admin then
    raise_application_error(-20001, 'Scott is NOT an administrator');
  end if;
end;

2.5 REMOVE_USER_ROLE Procedure Signature 1

This procedure removes an assigned role from a user.

Syntax

APEX_ACL.REMOVE_USER_ROLE ( p_application_id in number default wwv_flow_security.g_flow_id, p_user_name in varchar2, p_role_id in number );

Parameters

Table 2-5  REMOVE_USER_ROLE Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID from which you want to remove an assigned role from a user. Defaults to the current application.</td>
</tr>
</tbody>
</table>
Table 2-5  (Cont.) REMOVE_USER_ROLE Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to remove the role from.</td>
</tr>
<tr>
<td>p_role_id</td>
<td>The ID of the role.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use REMOVE_USER_ROLE procedure to remove role ID of 2505704029884282 from the user name called 'SCOTT' in application 255.

```sql
begin
    APEX_ACL.REMOVE_USER_ROLE (|
        p_application_id => 255,|
        p_user_name      => 'SCOTT',|
        p_role_id        => 2505704029884282 |
    );
end;
```

2.6 REMOVE_USER_ROLE Procedure Signature 2

This procedure removes an assigned role from a user.

Syntax

```sql
begin
    APEX_ACL.REMOVE_USER_ROLE (|
        p_application_id => 255,|
        p_user_name => 'SCOTT',|
        p_role_static_id => 'ADMINISTRATOR' |
    );
end;
```

Parameters

Table 2-6  REMOVE_USER_ROLE Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID from which you want to remove an assigned role from a user. It defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to remove the role from.</td>
</tr>
<tr>
<td>p_role_static_id</td>
<td>The case insensitive name of the role static ID.</td>
</tr>
</tbody>
</table>
Example
The following example shows how to use REMOVE_USER_ROLE procedure to remove role static ID 'ADMINISTRATOR' from the user name 'SCOTT' in application 255.

```
begin
    APEX_ACL.REMOVE_USER_ROLE ( 
        p_application_id => 255,
        p_user_name => 'SCOTT',
        p_role_static_id => 'ADMINISTRATOR' );
end;
```

2.7 REPLACE_USER_ROLES Procedure Signature 1

This procedure replaces any existing assigned user roles to new array of roles.

Syntax

```
APEX_ACL.REPLACE_USER_ROLES ( 
    p_application_id in number   default wwv_flow_security.g_flow_id,
    p_user_name      in varchar2,
    p_role_ids       in wwv_flow_t_number );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to replace user role. Defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to replace the role.</td>
</tr>
<tr>
<td>p_role_ids</td>
<td>The array of NUMBER type role IDs.</td>
</tr>
</tbody>
</table>

Example
The following example shows how to use REPLACE_USER_ROLES procedure to replace existing roles to new role IDs of 2505704029884282, 345029884282 for the user name called 'SCOTT' in application 255.

```
begin
    APEX_ACL.REPLACE_USER_ROLES ( 
        p_application_id => 255,
        p_user_name => 'SCOTT',
        p_role_ids => wwv_flow_t_number( 2505704029884282,
            345029884282 ) );
end;
```
2.8 REPLACE_USER_ROLES Procedure Signature 2

This procedure replaces any existing assigned user roles to new array of roles.

Syntax

APEX_ACL.REPLACE_USER_ROLES ( 
    p_application_id in number   default wwv_flow_security.g_flow_id, 
    p_user_name       in varchar2, 
    p_role_static_ids in wwv_flow_t_varchar2 );

Parameters

Table 2-8  REPLACE_USER_ROLES Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to replace user role. Defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to replace the role.</td>
</tr>
<tr>
<td>p_role_static_ids</td>
<td>The array of case insensitive VARCHAR2 type of role static IDs.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use REPLACE_USER_ROLES procedure to replace existing roles to new role static IDs of 'ADMINISTRATOR' and 'CONTRIBUTOR' for the user name called 'SCOTT' in application 255.

begin
    APEX_ACL.REPLACE_USER_ROLES ( 
        p_application_id => 255, 
        p_user_name      => 'SCOTT', 
        p_role_static_ids => wwv_flow_t_varchar2( 'ADMINISTRATOR', 'CONTRIBUTOR' ) );
end;

2.9 REMOVE_ALL_USER_ROLES Procedure

This procedure removes all assigned roles from a user.

Syntax

APEX_ACL.REMOVE_ALL_USER_ROLES ( 
    p_application_id in number   default wwv_flow_security.g_flow_id, 
    p_user_name       in varchar2 );
Parameters

Table 2-9  REMOVE_ALL_USER_ROLES Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID for which you want to remove all assigned roles from a user. Defaults to the current application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The case insensitive name of the application user to remove all assigned roles.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use REMOVE_ALL_USER_ROLES procedure to removes all assigned roles from the user name called 'SCOTT' in application 255.

begin
  APEX_ACL.REMOVE_ALL_USER_ROLES {
    p_application_id  => 255,
    p_user_name       => 'SCOTT' );
  end;

APEX_APPLICATION_INSTALL

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

- Package Overview
- Attributes Manipulated by APEX_APPLICATION_INSTALL
- Import Script Examples
- CLEAR_ALL Procedure
- GENERATE_APPLICATION_ID Procedure
- GENERATE_OFFSET Procedure
- GET_APPLICATION_ALIAS Function
- GET_APPLICATION_ID Function
- GET_APPLICATION_NAME Function
- GET_AUTO_INSTALL_SUP_OBJ Function
- GET_IMAGE_PREFIX Function
- GET_KEEP_SESSIONS Function
- GET_NO_PROXY_DOMAINS Function
- GET_OFFSET Function
- GET_PROXY Function
- GET_REMOTE_SERVER_BASE_URL Function
- GET_REMOTE_SERVER_HTTPS_HOST Function
- GET_SCHEMA Function
- GET_WORKSPACE_ID Function
- SET_APPLICATION_ALIAS Procedure
- SET_APPLICATION_ID Procedure
- SET_APPLICATION_NAME Procedure
- SET_AUTO_INSTALL_SUP_OBJ Procedure
- SET_IMAGE_PREFIX Procedure
- SET_KEEP_SESSIONS Procedure
- SET_OFFSET Procedure
- SET_PROXY Procedure
- SET_REMOTE_SERVER Procedure
- SET_SCHEMA Procedure
- SET_WORKSPACE_ID Procedure
3.1 Package Overview

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

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

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

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

```
set serveroutput on unlimited
```

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

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

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

3.2 Attributes Manipulated by APEX_APPLICATION_INSTALL

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace ID</td>
<td>Workspace ID of the imported application. See GET_WORKSPACE_ID Function, SET_WORKSPACE_ID Procedure.</td>
</tr>
</tbody>
</table>
Table 3-1  (Cont.) Attributes Manipulated by the APEX_APPLICATION_INSTALL API

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>Offset value used during application import. See GENERATE_OFFSET Procedure, GET_OFFSET Function, SET_OFFSET Procedure.</td>
</tr>
<tr>
<td>Schema</td>
<td>The parsing schema (&quot;owner&quot;) of the imported application. See GET_SCHEMA Function, SET_SCHEMA Procedure.</td>
</tr>
<tr>
<td>Name</td>
<td>Application name of the imported application. See GET_APPLICATION_NAME Function, SET_APPLICATION_NAME Procedure.</td>
</tr>
<tr>
<td>Alias</td>
<td>Application alias of the imported application. See GET_APPLICATION_ALIAS Function, SET_APPLICATION_ALIAS Procedure.</td>
</tr>
<tr>
<td>Image Prefix</td>
<td>The image prefix of the imported application. See GET_IMAGE_PREFIX Function, SET_IMAGE_PREFIX Procedure.</td>
</tr>
<tr>
<td>Proxy</td>
<td>The proxy server attributes of the imported application. See GET_PROXY Function, SET_PROXY Procedure.</td>
</tr>
</tbody>
</table>

3.3 Import Script Examples

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

Import Application without Modification

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

```sql
@f645.sql
```

Import Application with Specified Application ID

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

```sql
begin
    apex_application_install.set_application_id( 702);
    apex_application_install.generate_offset;
    apex_application_install.set_application_alias( 'F' || apex_application_install.get_application_id );
end;
```
Import Application with Generated Application ID

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

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

Import Application into Different Workspace using Different Schema

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

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

Import into Training Instance for Three Different Workspaces

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

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

@645.sql

@f645.sql
3.4 CLEAR_ALL Procedure

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

Syntax

APEX_APPLICATION_INSTALL.CLEAR_ALL;

Parameters

None.

Example

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

begin
    apex_application_install.clear_all;
end;

3.5 GENERATE_APPLICATION_ID Procedure

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

APEX_APPLICATION_INSTALL.GENERATE_APPLICATION_ID;

Parameters

None.

Example

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

See Also:

• "SET_APPLICATION_ID Procedure"
• "GET_APPLICATION_ID Function"

3.6 GENERATE_OFFSET Procedure

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

Syntax

APEX_APPLICATION_INSTALL.GENERATE_OFFSET;

Parameters

None.

Example

For examples of this procedure call, see "Import Application with Specified Application ID", "Import Application with Generated Application ID", and "Import into Training Instance for Three Different Workspaces".

See Also:

• "GET_OFFSET Function"
• "SET_OFFSET Procedure"
3.7 GET_APPLICATION_ALIAS Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_APPLICATION_ALIAS
RETURN VARCHAR2;

Parameters

None.

Example

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

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

See Also:

"SET_APPLICATION_ALIAS Procedure"

3.8 GET_APPLICATION_ID Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_APPLICATION_ID
RETURN NUMBER;

Parameters

None.
Example

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

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

See Also:

- "SET_APPLICATION_ID Procedure"
- "GENERATE_APPLICATION_ID Procedure"

3.9 GET_APPLICATION_NAME Function

This function gets the application name of the import application.

Syntax

APEX_APPLICATION_INSTALL.GET_APPLICATION_NAME
RETURN VARCHAR2;

Parameters

None.

Example

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

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

See Also:

"SET_APPLICATION_NAME Procedure"
3.10 GET_AUTO_INSTALL_SUP_OBJ Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_AUTO_INSTALL_SUP_OBJ
RETURN BOOLEAN;

Parameters

None.

Example

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

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

3.11 GET_IMAGE_PREFIX Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_IMAGE_PREFIX
RETURN VARCHAR2;

Parameters

None.

Example

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

```sql
declare
   l_image_prefix varchar2(255);
begin
   l_image_prefix := apex_application_install.get_image_prefix;
end;
```
3.12 GET_KEEP_SESSIONS Function

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

Syntax

```plaintext
function get_keep_sessions
    return boolean
end;
```

Example

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

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

See Also:

"SET_KEEP_SESSIONS Procedure"

3.13 GET_NO_PROXY_DOMAINS Function

Use this function to get the No Proxy Domains attribute of an application to be imported.

Syntax

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

Parameters

None.
Example

declare
    l_no_proxy_domains varchar2(255);
begin
    l_no_proxy_domains := apex_application_install.get_no_proxy_domains;
end;

See Also:

"SET_PROXY Procedure"

3.14 GET_OFFSET Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_OFFSET
RETURN NUMBER;

Parameters

None.

Example

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

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

See Also:

• "SET_OFFSET Procedure"
• "GENERATE_OFFSET Procedure"
### 3.15 GET_PROXY Function

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

**Syntax**

APEX_APPLICATION_INSTALL.GET_PROXY
RETURN VARCHAR2;

**Parameters**

None.

**Example**

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

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

**See Also:**

"SET_PROXY Procedure"

---

### 3.16 GET_REMOTE_SERVER_BASE_URL Function

Use this function to get the Base URL property to be used for a given remote server during application import.

**Syntax**

APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_BASE_URL(
    p_static_id IN VARCHAR2)
RETURN VARCHAR2;

**Parameters**

**Table 3-2** GET_REMOTE_SERVER_BASE_URL Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_static_id</td>
<td>Static ID to reference the remote server object.</td>
</tr>
</tbody>
</table>
Example

declare
   l_base_url varchar2(255);
begin
   l_base_url :=
apex_application_install.get_remote_server_base_url( 'MY_REMOTE_SERVER' );
end;

See Also:
"SET_REMOTE_SERVER Procedure"

3.17 GET_REMOTE_SERVER_HTTPS_HOST Function

Use this function to get the HTTPS Host property to be used for a given remote server during application import.

Syntax

APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_HTTPS_HOST(
   p_static_id IN VARCHAR2)
RETURN VARCHAR2;

Parameters

Table 3-3  GET_REMOTE_SERVER_HTTPS_HOST Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_static_id</td>
<td>Static ID to reference the remote server object.</td>
</tr>
</tbody>
</table>

Example

declare
   l_https_host varchar2(255);
begin
   l_https_host :=
apex_application_install.get_remote_server_https_host( 'MY_REMOTE_SERVER' )
   ;
end;
3.18 GET_SCHEMA Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_SCHEMA
RETURN VARCHAR2;

Parameters

None.

Example

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

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

3.19 GET_WORKSPACE_ID Function

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

Syntax

APEX_APPLICATION_INSTALL.GET_WORKSPACE_ID
RETURN NUMBER;

Parameters

None.
Example

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

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

See Also:

"SET_WORKSPACE_ID Procedure"

3.20 SET_APPLICATION_ALIAS Procedure

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

Syntax

APEX_APPLICATION_INSTALL.SET_APPLICATION_ALIAS(
p_application_alias IN VARCHAR2);

Parameters

Table 3-4  SET_APPLICATION_ALIAS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_alias</td>
<td>The application alias. The application alias is an alphanumeric identifier. It cannot exceed 255 characters, must unique within a workspace and, ideally, is unique within an entire instance.</td>
</tr>
</tbody>
</table>

Example

For examples of this procedure call, see "Import Application with Specified Application ID,""Import Application with Generated Application ID,""Import Application into Different Workspace using Different Schema" and "Import into Training Instance for Three Different Workspaces."

See Also:

"GET_APPLICATION_ALIAS Function"
3.21 SET_APPLICATION_ID Procedure

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

Syntax

APEX_APPLICATION_INSTALL.SET_APPLICATION_ID (p_application_id IN NUMBER);

Parameters

Table 3-5  SET_APPLICATION_ID Parameters

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

Example

For an example of this procedure call, see "Import Application with Specified Application ID."

See Also:

- "SET_APPLICATION_ID Procedure"
- "GENERATE_APPLICATION_ID Procedure"

3.22 SET_APPLICATION_NAME Procedure

This procedure sets the application name of the import application.

Syntax

APEX_APPLICATION_INSTALL.SET_APPLICATION_NAME; (p_application_name IN VARCHAR2);
Parameters

Table 3-6 SET_APPLICATION_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_name</td>
<td>This is the application name. The application name cannot be null and cannot be longer than 255 characters.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

"GET_APPLICATION_NAME Function"

3.23 SET_AUTO_INSTALL_SUP_OBJ Procedure

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

Syntax

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

Parameters

Table 3-7 SET_AUTO_INSTALL_SUP_OBJ Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_auto_install_sup_obj</td>
<td>The automatic install of supporting objects Boolean value.</td>
</tr>
</tbody>
</table>
Example
The following example gets the automatic install of supporting objects setting. If it is not set to install automatically, it sets to `true` to override export file settings of automatic install of supporting objects.

```
begin

    apex_application_install.set_auto_install_sup_obj( p_auto_install_sup_obj => true );

end;
```

### 3.24 SET_IMAGE_PREFIX Procedure

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

**Syntax**

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

**Parameters**

**Table 3-8**  SET_AUTO_INSTALL_SUP_OBJ Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_auto_install_sup_obj</td>
<td>The automatic install of supporting objects Boolean value.</td>
</tr>
</tbody>
</table>

**Example**

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

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

*See Also:*

"GET_IMAGE_PREFIX Function"
3.25 SET_KEEP_SESSIONS Procedure

This procedure preserves sessions associated with the application on upgrades.

Syntax

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

Parameters

Table 3-9  SET_KEEP_SESSIONS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_keep_sessions</td>
<td>false is the default value. true if sessions should be preserved, false if</td>
</tr>
<tr>
<td></td>
<td>they should be deleted. KEEP_SESSIONS_ON_UPGRADE controls the default behavior.</td>
</tr>
<tr>
<td></td>
<td>If it is N (the default), sessions will be deleted. KEEP_SESSIONS_ON_UPGRADE</td>
</tr>
<tr>
<td></td>
<td>is an instance parameter.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

"GET_KEEP_SESSIONS Function"

3.26 SET_OFFSET Procedure

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

Syntax

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

Table 3-10  SET_OFFSET Parameters

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

Example

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

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

See Also:
- "GET_OFFSET Function"
- "GENERATE_OFFSET Procedure"

3.27 SET_PROXY Procedure

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

Syntax

APEX_APPLICATION_INSTALL.SET_PROXY (  
  p_proxy       IN VARCHAR2,  
  p_no_proxy_domains IN VARCHAR2 DEFAULT NULL );
Parameters

Table 3-11  SET_PROXY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_proxy</td>
<td>The proxy server. There is no default value. The proxy server cannot be more than 255 characters and should not include any protocol prefix such as http://. A sample value might be: www-proxy.company.com</td>
</tr>
<tr>
<td>p_no_proxy_domains</td>
<td>The list of domains for which the proxy server should not be used. There is no default value.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

"SET_PROXY Procedure"

3.28 SET_REMOTE_SERVER Procedure

Use this procedure to set the Base URL and the HTTPS Host attributes for remote servers of the imported application. Remote Servers are identified by their Static ID.

Syntax

```
APEX_APPLICATION_INSTALL.SET_REMOTE_SERVER(
    p_static_id IN VARCHAR2,
    p_base_url IN VARCHAR2,
    p_https_host IN VARCHAR2 default null );
```

Parameters

Table 3-12  SET_REMOTE_SERVER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_static_id</td>
<td>Static ID to reference the remote server object.</td>
</tr>
<tr>
<td>p_base_url</td>
<td>New Base URL to use for this remote server object.</td>
</tr>
</tbody>
</table>
Table 3-12  (Cont.) SET_REMOTE_SERVER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_https_host</td>
<td>New HTTPS Host Property to use for this remote server object. Only relevant when the base URL is https:// and the database version is 12.2 or greater.</td>
</tr>
</tbody>
</table>

Example

```
begin
    apex_application_install.set_remote_server(
        p_static_id => 'MY_REMOTE_SERVER',
        p_base_url => 'http://production.company.com'
    );
end;
```

See Also:

"GET_REMOTE_SERVER_BASE_URL Function","GET_REMOTE_SERVER_HTTPS_HOST Function"

3.29 SET_SCHEMA Procedure

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

Syntax

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

Parameters

Table 3-13  SET_SCHEMA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_schema</td>
<td>The schema name.</td>
</tr>
</tbody>
</table>

Example

For examples of this procedure call, see "Import Application into Different Workspace using Different Schema" and "Import into Training Instance for Three Different Workspaces".
3.30 SET_WORKSPACE_ID Procedure

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

Syntax

APEX_APPLICATION_INSTALL.SET_WORKSPACE_ID (p_workspace_id IN NUMBER);

Parameters

Table 3-14 SET_WORKSPACE_ID Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The workspace ID.</td>
</tr>
</tbody>
</table>

Example

For examples of this procedure call, see "Import Application into Different Workspace using Different Schema" and "Import into Training Instance for Three Different Workspaces".

3.31 SET_WORKSPACE_Procedure

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

Syntax

procedure set_workspace (p_workspace in varchar2 );
Parameters

Table 3-15  SET_WORKSPACE_Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The workspace name.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "GET_WORKSPACE_ID Function"
- "SET_WORKSPACE_ID Procedure"
The APEX_AUTHENTICATION package provides a public API for authentication plug-in.

- Constants
- CALLBACK Procedure
- CALLBACK 1 Procedure
- CALLBACK 2 Procedure
- GET_CALLBACK_URL Function
- GET_LOGIN_USERNAME_COOKIE Function
- IS_AUTHENTICATED Function
- IS_PUBLIC_USER Function
- LOGIN Procedure
- LOGOUT Procedure
- POST_LOGIN Procedure
- SEND_LOGIN_USERNAME_COOKIE Procedure

4.1 Constants

The following constant is used by this package.

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

4.2 CALLBACK Procedure

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

**Syntax**

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

Table 4-1  APEX_AUTHENTICATION.CALLBACK Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_session_id</td>
<td>The Application Express session identifier.</td>
</tr>
<tr>
<td>p_app_id</td>
<td>The database application identifier.</td>
</tr>
<tr>
<td>p_page_id</td>
<td>Optional page identifier.</td>
</tr>
<tr>
<td>p_ajax_identifierp</td>
<td>The system generated Ajax identifier. See &quot;GET_AJAX_IDENTIFIER Function.&quot;</td>
</tr>
<tr>
<td>p_x01 through p_x10</td>
<td>Optional parameters that the external login passes to the authentication plugin.</td>
</tr>
</tbody>
</table>

Example 1

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

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

Example 2

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

```
---- create or replace package body my_custom_sso as
procedure login (  
    p_on_success in varchar2 )  
    is
    l_login_id varchar2(32);
begin
    l_login_id := rawtohex(sys.dbms_crypto.random(32));
    insert into login_data(id, username) values (l_login_id, 'JOE USER');
    sys.owa_util.redirect_url (  
```
4.3 CALLBACK 1 Procedure

This procedure is the landing resource for OAuth2-based authentication schemes. The parameters are defined by the OAuth2 spec. This procedure gets called via redirects, by external authentication providers.

**Syntax**

```sql
procedure callback (  
  state             in varchar2,  
  code              in varchar2 default null,  
  error             in varchar2 default null,  
  error_description in varchar2 default null,  
  error_uri         in varchar2 default null,  
  error_reason      in varchar2 default null,  
  error_code        in varchar2 default null,  
  error_message     in varchar2 default null,  
  authuser          in varchar2 default null,  
  session_state     in varchar2 default null,  
  prompt            in varchar2 default null,  
  scope             in varchar2 default null );
```

4.4 CALLBACK 2 Procedure

This procedure is an alternative to Callback 1.

**Syntax**

```sql
procedure callback2 (  
  p_session_id      in number,  
  p_app_id          in number,  
  p_ajax_identifier in varchar2,  
  p_page_id         in number   default null,  
  p_x01             in varchar2 default null,  
  p_x02             in varchar2 default null,  
  p_x03             in varchar2 default null,  
  p_x04             in varchar2 default null,  
  p_x05             in varchar2 default null,  
  p_x06             in varchar2 default null,  
  p_x07             in varchar2 default null,  
```

---

Note:

"GET_CALLBACK_URL Function"
"CALLBACK 2 Procedure"
4.5 GET_CALLBACK_URL Function

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_x01 through p_x10</td>
<td>Optional parameters that the external login passes to the authentication plugin.</td>
</tr>
<tr>
<td>p_callback_name</td>
<td>Optional public name of the callback, defaults to apex_authentication.callback.</td>
</tr>
</tbody>
</table>
4.6 GET_LOGIN_USERNAME_COOKIE Function

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

Syntax

```sql
get_login_username_cookie (  
    p_cookie_name in varchar2 default c_default_username_cookie )  
return varchar2;
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_cookie_name</td>
<td>The cookie name which stores the username in the browser.</td>
</tr>
</tbody>
</table>

Example

The example code below could be from a Before Header process. It populates a text item `P101_USERNAME` with the cookie value and a switch `P101_REMEMBER_USERNAME`, based on whether the cookie already has a value.

```sql
:P101_USERNAME          := apex_authentication.get_login_username_cookie;
:P101_REMEMBER_USERNAME := case when :P101_USERNAME is not null  
                              then 'Y'  
                              else 'N'  
                          end;
```

See Also:

"CALLBACK Procedure"
4.7 IS_AUTHENTICATED Function

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

Syntax

APEX_AUTHENTICATION.IS_AUTHENTICATED
RETURN BOOLEAN;

Parameters

None.

Example

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

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

Note:

"IS_PUBLIC_USER Function"

4.8 IS_PUBLIC_USER Function

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

Syntax

APEX_AUTHENTICATION.IS_PUBLIC_USER
return BOOLEAN;

Parameters

None.
Example

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

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

### 4.9 LOGIN Procedure

This procedure authenticates the user in the current session.

Login processing has the following steps:

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

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>The user's name.</td>
</tr>
<tr>
<td>p_password</td>
<td>The user's password.</td>
</tr>
<tr>
<td>p_uppercase_username</td>
<td>If TRUE then p_username is converted to uppercase.</td>
</tr>
</tbody>
</table>
Example

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

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

Note:

"POST_LOGIN Procedure"

4.10 LOGOUT Procedure

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

Syntax

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

Parameters

**Table 4-5  APEX_AUTHENTICATION.LOGOUT Procedure Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_session_id</td>
<td>The Application Express session identifier of the session to close.</td>
</tr>
<tr>
<td>p_app_id</td>
<td>The database application identifier.</td>
</tr>
<tr>
<td>p_ws_app_id</td>
<td>The websheet application identifier.</td>
</tr>
</tbody>
</table>

Example

This example logs the session out.

apex_authentication.logout(:SESSION, :APP_ID);

4.11 POST_LOGIN Procedure

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

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

Parameters

Table 4-6  APEX_AUTHENTICATION.POST_LOGIN Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>The user's name.</td>
</tr>
<tr>
<td>p_password</td>
<td>The user's password.</td>
</tr>
<tr>
<td>p_uppercase_username</td>
<td>If TRUE then p_username is converted to uppercase.</td>
</tr>
</tbody>
</table>

Example

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

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

Note:

"LOGIN Procedure"

4.12 SEND_LOGIN_USERNAME_COOKIE Procedure

This procedure sends a cookie with the username.

Syntax

send_login_username_cookie(
  p_username in varchar2,
  p_cookie_name in varchar2 default c_default_username_cookie,
  p_consent in boolean default false
);

Parameters

Table 4-7  APEX_AUTHENTICATION.SEND_LOGIN_USERNAME_COOKIE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>The user's name.</td>
</tr>
</tbody>
</table>
Table 4-7  (Cont.)
APEX_AUTHENTICATION.SEND_LOGIN_USERNAME_COOKIE Procedure

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_cookie_name</td>
<td>The cookie name which stores p_username in the browser.</td>
</tr>
<tr>
<td>p_consent</td>
<td>Control if the cookie should actually be sent. If true, assume the user gave consent to send the cookie. If false, do not send the cookie. If there is no consent and the cookie already exists, the procedure overwrites the existing cookie value with null.</td>
</tr>
</tbody>
</table>

Example

The example code below could be from a page submit process on a login page, which saves the username in a cookie when consent is given. P101_REMEMBER_USERNAME could be a switch. On rendering, it could be set to 'Y' when the cookie has a value.

```sql
apex_authentication.send_login_username_cookie (
  p_username => :P101_USERNAME,
  p_consent  => :P101_REMEMBER_USERNAME = 'Y');
```

See Also:

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

- ENABLE_DYNAMIC_GROUPS Procedure
- IS_AUTHORIZED Function
- RESET_CACHE Procedure

### 5.1 ENABLE_DYNAMIC_GROUPS Procedure

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

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

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

**Syntax**

APEX_AUTHORIZATION.ENABLE_DYNAMIC_GROUPS (  
    p_group_names IN apex_t_varchar2 );

**Parameters**

**Table 5-1  ENABLE_DYNAMIC_GROUPS Procedure Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_names</td>
<td>Table of group names.</td>
</tr>
</tbody>
</table>

**Example**

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

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

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

See Also:

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

Syntax

APEX_AUTHORIZATION.IS_AUTHORIZED ( p_authorization_name IN VARCHAR2 ) RETURN BOOLEAN;

Parameters

Table 5-2  IS_AUTHORIZED Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_authorization_name</td>
<td>The name of an authorization scheme in the application.</td>
</tr>
</tbody>
</table>

Returns

Table 5-3  IS_AUTHORIZED Function Returns

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>If the authorization is successful.</td>
</tr>
<tr>
<td>FALSE</td>
<td>If the authorization is not successful.</td>
</tr>
</tbody>
</table>

Example

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

```sql
begin
    sys.htp.p('User Is Admin: ' ||
               case apex_authorization.is_authorized (  
```
5.3 RESET_CACHE Procedure

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

Syntax

APEX_AUTHORIZATION.RESET_CACHE;

Parameters

None.

Example

This examples resets the authorization cache.

apex_authorization.reset_cache;
The APEX_APP_SETTING package provides utilities you can use when programming in the Oracle Application Express environment related to application setting shared components. You can use the APEX_APP_SETTING package to get and set the value of application settings.

- **SET_VALUE Procedure**
- **GET_VALUE Function**

### 6.1 SET_VALUE Procedure

This procedure changes the application setting value in the current application.

**Syntax**

APEX_APP_SETTING.SET_VALUE(
    p_name     in varchar2,
    p_value    in varchar2);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| p_name     | The case insensitive name of the application setting. An error raised if:  
     • Application Setting name does not exist.  
     • If build option associated with application setting is disabled. |
| p_value    | The value of the application setting. An error raised if:  
     • The value is set to required, but null value passed.  
     • The valid values defined, but the value is not in one of the valid values. |

**Example**

The following example shows how to use the SET_VALUE procedure to set the value of application setting `ACCESS_CONTROL_ENABLED`.

```sql
begin
    APEX_APP_SETTING.SET_VALUE(
        p_name  => 'ACCESS_CONTROL_ENABLED',
        p_value => 'Y');
end;
```
6.2 GET_VALUE Function

This function gets the application setting value in the current application.

Syntax

APEX_APP_SETTING.GET_VALUE(  
    p_name        in varchar2 ) return varchar2;

Parameters

Table 6-2  GET_VALUE Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| p_name     | The case insensitive name of the application setting. An error raised if:  
  • Application Setting name does not exist.  
  • If build option, associated with application setting is disabled. |

Example

The following example shows how to use the GET_VALUE function to retrieve the value of application setting ACCESS_CONTROL_ENABLED.

```
declare  
  l_value varchar2(4000);  
begin  
  l_value := APEX_APP_SETTING.GET_VALUE( p_name =>  
    'ACCESS_CONTROL_ENABLED');
end;
```
APEX_COLLECTION

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

• About the APEX_COLLECTION API
• Naming Collections
• Creating a Collection
• About the Parameter p_generate_md5
• Accessing a Collection
• Merging Collections
• Truncating a Collection
• Deleting a Collection
• Deleting All Collections for the Current Application
• Deleting All Collections in the Current Session
• Adding Members to a Collection
• About the Parameters p_generate_md5, p_clob001, p_blob001, and p_xmltype001
• Updating Collection Members
• Deleting Collection Members
• Obtaining a Member Count
• Resequencing a Collection
• Verifying Whether a Collection Exists
• Adjusting a Member Sequence ID
• Sorting Collection Members
• Clearing Collection Session State
• Determining Collection Status
• ADD_MEMBER Procedure
• ADD_MEMBER Function
• ADD_MEMBERS Procedure
• COLLECTION_EXISTS Function
• COLLECTION_HAS_CHANGED Function
• COLLECTION_MEMBER_COUNT Function
• CREATE_COLLECTION Procedure
7.1 About the APEX_COLLECTION API

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

The following are examples of when you might use collections:
• When you are creating a data-entry wizard in which multiple rows of information first need to be collected within a logical transaction. You can use collections to temporarily store the contents of the multiple rows of information, before performing the final step in the wizard when both the physical and logical transactions are completed.

• When your application includes an update page on which a user updates multiple detail rows on one page. The user can make many updates, apply these updates to a collection and then call a final process to apply the changes to the database.

• When you are building a wizard where you are collecting an arbitrary number of attributes. At the end of the wizard, the user then performs a task that takes the information temporarily stored in the collection and applies it to the database.

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

7.2 Naming Collections

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

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

See Also:

• “Accessing a Collection”
• “CREATE_COLLECTION Procedure”
• “CREATE_OR_TRUNCATE_COLLECTION Procedure”

7.3 Creating a Collection

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

You use the following methods to create a collection:

• CREATE_COLLECTION

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

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

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

- **CREATE_COLLECTION_FROM_QUERY2**
  This method creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. It is identical to the `CREATE_COLLECTION_FROM_QUERY`, however, the first 5 columns of the `SELECT` clause must be numeric. After the numeric columns, there can be up to 50 character columns in the `SELECT` clause.

- **CREATE_COLLECTION_FROM_QUERY_B**
  This method offers significantly faster performance than the `CREATE_COLLECTION_FROM_QUERY` method by performing bulk SQL operations, but has the following limitations:
  - No column value in the select list of the query can be more than 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error is raised during execution.
  - The MD5 checksum is not computed for any members in the collection.

- **CREATE_COLLECTION_FROM_QUERYB2**
  This method also creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. It is identical to the `CREATE_COLLECTION_FROM_QUERY_B`, however, the first five columns of the `SELECT` clause must be numeric. After the numeric columns, there can be up to 50 character columns in the `SELECT` clause.

### See Also:
- "CREATE_COLLECTION Procedure"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure"

### 7.4 About the Parameter p_generate_md5

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

See Also:

- "Determining Collection Status" for information about using the GET_MEMBER_MD5 function
- "GET_MEMBER_MD5 Function"

7.5 Accessing a Collection

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

<table>
<thead>
<tr>
<th>COLLECTION_NAME</th>
<th>NOT NULL VARCHAR2(255)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ_ID</td>
<td>NOT NULL NUMBER</td>
</tr>
<tr>
<td>C001</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>C002</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>C003</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>C004</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>C005</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>C050</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>N001</td>
<td>NUMBER</td>
</tr>
<tr>
<td>N002</td>
<td>NUMBER</td>
</tr>
<tr>
<td>N003</td>
<td>NUMBER</td>
</tr>
<tr>
<td>N004</td>
<td>NUMBER</td>
</tr>
<tr>
<td>N005</td>
<td>NUMBER</td>
</tr>
<tr>
<td>D001</td>
<td>DATE</td>
</tr>
<tr>
<td>D002</td>
<td>DATE</td>
</tr>
<tr>
<td>D003</td>
<td>DATE</td>
</tr>
<tr>
<td>D004</td>
<td>DATE</td>
</tr>
<tr>
<td>D005</td>
<td>DATE</td>
</tr>
<tr>
<td>CLOB001</td>
<td>CLOB</td>
</tr>
<tr>
<td>BLOB001</td>
<td>BLOB</td>
</tr>
<tr>
<td>XMLTYPE001</td>
<td>XMLTYPE</td>
</tr>
<tr>
<td>MD5_ORIGINAL</td>
<td>VARCHAR2(4000)</td>
</tr>
</tbody>
</table>

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

```sql
SELECT c001, c002, c003, n001, d001, clob001
FROM APEX_collections
WHERE collection_name = 'DEPARTMENTS'
```
7.6 Merging Collections

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

See Also:

"MERGE_MEMBERS Procedure"

7.7 Truncating a Collection

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

See Also:

"TRUNCATE_COLLECTION Procedure"

7.8 Deleting a Collection

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

See Also:

"DELETE_COLLECTION Procedure"

7.9 Deleting All Collections for the Current Application

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

See Also:

"DELETE_ALL_COLLECTIONS Procedure"
7.10 Deleting All Collections in the Current Session

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

See Also:

- "DELETE_ALL_COLLECTIONS_SESSION Procedure"

7.11 Adding Members to a Collection

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

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

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

See Also:

- "ADD_MEMBER Procedure"
- "ADD_MEMBER Function"
- "ADD_MEMBERS Procedure"

7.12 About the Parameters `p_generate_md5`, `p_clob001`, `p_blob001`, and `p_xmltype001`

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

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

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

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

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

---

7.14 Deleting Collection Members

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

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

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

See Also:
- "DELETE_MEMBER Procedure"
- "DELETE_MEMBERS Procedure"

7.16 Resequencing a Collection

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

See Also:
- "RESEQUENCE_COLLECTION Procedure"

7.17 Verifying Whether a Collection Exists

Use `COLLECTION_EXISTS` to determine if a collection exists.

See Also:
- "COLLECTION_EXISTS Function"

7.18 Adjusting a Member Sequence ID

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

### See Also:

- "MOVE_MEMBER_DOWN Procedure"
- "MOVE_MEMBER_UP Procedure"

#### 7.19 Sorting Collection Members

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

### See Also:

"SORT_MEMBERS Procedure"

#### 7.20 Clearing Collection Session State

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

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

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

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

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

### See Also:

"TRUNCATE_COLLECTION Procedure"
7.21 Determining Collection Status

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

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

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

When you add a new member to a collection, an MD5 message digest is computed against all 50 attributes and the CLOB attribute if the p_generated_md5 parameter is set to `YES`. You can access this value from the `MD5ORIGINAL` column of the view `APEX_COLLECTION`. You can access the MD5 message digest for the current value of a specified collection member by using the function `GET_MEMBER_MD5`.

See Also:

- “`RESET_COLLECTION_CHANGED Procedure`”
- “`COLLECTION_HAS_CHANGED Function`”
- “`GET_MEMBER_MD5 Function`”

7.22 ADD_MEMBER Procedure

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

Syntax

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

Parameters

![Note:]
Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.

Table 7-1 ADD_MEMBER Procedure Parameters

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

Example

The following is an example of the ADD_MEMBER procedure.

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

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

Syntax

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

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
### Table 7-2  ADD_MEMBER Function Parameters

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

#### Example

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

#### See Also:
- "GET_MEMBER_MD5 Function"
- "ADD_MEMBER Procedure"
- "ADD_MEMBERS Procedure"

### 7.24 ADD_MEMBERS Procedure

Use this procedure to add an array of members to a collection. An error is raised if the specified collection does not exist for the current user in the same session for the
current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) adds the new member with a sequence ID of 9. The count of elements in the p_c001 PL/SQL table is used as the total number of items across all PL/SQL tables. For example, if p_c001.count is 2 and p_c002.count is 10, only 2 members are added. If p_c001 is null an application error is raised.

Syntax

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

Parameters

Note:

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

Table 7-3 ADD_MEMBERS Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.</td>
</tr>
<tr>
<td>p_c001 through p_c050</td>
<td>Array of character attribute values to be added.</td>
</tr>
<tr>
<td>p_n001 through p_n005</td>
<td>Array of numeric attribute values to be added.</td>
</tr>
<tr>
<td>p_d001 through p_d005</td>
<td>Array of date attribute values to be added.</td>
</tr>
</tbody>
</table>
Table 7-3  (Cont.) ADD_MEMBERS Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_generate_md5</td>
<td>Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.</td>
</tr>
</tbody>
</table>

Example

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

```sql
Begin
  APEX_COLLECTION.ADD_MEMBERS(
    p_collection_name => 'EMPLOYEE',
    p_c001 => l_arr1,
    p_c002 => l_arr2);
End;
```

See Also:

- "GET_MEMBER_MD5 Function"
- "ADD_MEMBER Procedure"
- "ADD_MEMBER Function"

7.25 COLLECTION_EXISTS Function

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

Syntax

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

Table 7-4  COLLECTION_EXISTS Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. Maximum length is 255 bytes. The collection name is not case sensitive and is converted to upper case.</td>
</tr>
</tbody>
</table>

Example

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

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

7.26 COLLECTION_HAS_CHANGED Function

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

Syntax

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

Parameters

Table 7-5  COLLECTION_HAS_CHANGED Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
</tbody>
</table>

Example

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

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

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

Syntax

APEX_COLLECTION.COLLECTION_MEMBER_COUNT ( 
    p_collection_name IN VARCHAR2) 
RETURN NUMBER;

Parameters

Table 7-6  COLLECTION_MEMBER_COUNT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection.</td>
</tr>
</tbody>
</table>

Example

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

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

7.28 CREATE_COLLECTION Procedure

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

Syntax

APEX_COLLECTION.CREATE_COLLECTION( 
    p_collection_name      IN VARCHAR2, 
    p_truncate_if_exists IN VARCHAR2 default 'NO');
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_truncate_if_exists</td>
<td>If YES, then members of the collection will first be truncated if the collection exists and no error will be raised. If NO (or not YES), and the collection exists, an error will be raised.</td>
</tr>
</tbody>
</table>

### Example

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

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

### See Also:

- "CREATE_OR_TRUNCATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure"

### 7.29 CREATE_OR_TRUNCATE_COLLECTION Procedure

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

### Syntax

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

Table 7-8  CREATE_OR_TRUNCATE_COLLECTION Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. The maximum length is 255 characters. All members of the named collection are removed if the named collection exists for the current user in the current session.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "CREATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure"

7.30 CREATE_COLLECTION_FROM_QUERY Procedure

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

Syntax

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

Table 7-9  CREATE_COLLECTION_FROM_QUERY Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_query</td>
<td>Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.</td>
</tr>
<tr>
<td>p_generate_md5</td>
<td>Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.</td>
</tr>
<tr>
<td>p_truncate_if_exists</td>
<td>If YES, then members of the collection will first be truncated if the collection exists and no error will be raised. If NO (or not YES), and the collection exists, an error will be raised.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "GET_MEMBER_MD5 Function"
- "CREATE_COLLECTION Procedure"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure"
- "CREATE_COLLECTION_FROMQUERYB2 Procedure"
7.31 CREATE_COLLECTION_FROM_QUERY2 Procedure

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

Syntax

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

Parameters

Table 7-10  CREATE_COLLECTION_FROM_QUERY2 Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_query</td>
<td>Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.</td>
</tr>
<tr>
<td>p_generate_md5</td>
<td>Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.</td>
</tr>
<tr>
<td>p_truncate_if_exists</td>
<td>If YES, then members of the collection will first be truncated if the collection exists and no error will be raised. If NO (or not YES), and the collection exists, an error will be raised.</td>
</tr>
</tbody>
</table>

Example

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

```
begin;
  APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY2 (  
    p_collection_name => 'EMPLOYEE',  
    p_query => 'select empno, sal, comm, deptno, null, hiredate, null, null, null, null, ename, job, mgr from emp',
```


7.32 CREATE_COLLECTION_FROM_QUERY_B Procedure

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

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

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

Syntax

APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B (  
  p_collection_name IN VARCHAR2,  
  p_query IN VARCHAR2,  
  p_names IN apex_application_global.vc_arr2,  
  p_values IN apex_application_global.vc_arr2,  
  p_max_row_count IN NUMBER default null,  
  p_truncate_if_exists IN VARCHAR2 default 'NO');
### Parameters

**Table 7-11 CREATE_COLLECTION_FROM_QUERY_B Procedure Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_query</td>
<td>Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.</td>
</tr>
<tr>
<td>p_names</td>
<td>Array of bind variable names used in the query statement.</td>
</tr>
<tr>
<td>p_values</td>
<td>Array of bind variable values used in the bind variables in the query statement.</td>
</tr>
<tr>
<td>p_max_row_count</td>
<td>Maximum number of rows returned from the query in p_query which should be added to the collection.</td>
</tr>
<tr>
<td>p_truncate_if_exists</td>
<td>If YES, then members of the collection will first be truncated if the collection exists and no error will be raised. If NO (or not YES), and the collection exists, an error will be raised.</td>
</tr>
</tbody>
</table>

### Example

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

```sql
declare
    l_query varchar2(4000);
Begin
    l_query := 'select empno, ename, job, sal from emp where deptno = :b1';
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B (p_collection_name => 'EMPLOYEES',
                                                      p_query => l_query,
                                                      p_names => apex_util.string_to_table('b1'),
                                                      p_values => apex_util.string_to_table('10'));
End;
```

### See Also:

- "GET_MEMBER_MD5 Function"
- "CREATE_COLLECTION Procedure"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure"
7.33 CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version)

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

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

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

Syntax

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

Parameters

Table 7-12 CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version) Parameters

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

Example

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

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

APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B
{
    p_collection_name => 'EMPLOYEES',
    p_query => l_query );
End;

See Also:

- "GET_MEMBER_MD5 Function"
- "CREATE_COLLECTION Procedure"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERYB2 Procedure"

7.34 CREATE_COLLECTION_FROM_QUERYB2 Procedure

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

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

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

Syntax

APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_names IN apex_application_global.vc_arr2,
    p_values IN apex_application_global.vc_arr2,
    p_max_row_count IN NUMBER default null,
    p_truncate_if_exists IN VARCHAR2 default 'NO');
Parameters

Table 7-13  CREATE_COLLECTION_FROM_QUERYB2 Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_query</td>
<td>Query to execute to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.</td>
</tr>
<tr>
<td>p_names</td>
<td>Array of bind variable names used in the query statement.</td>
</tr>
<tr>
<td>p_values</td>
<td>Array of bind variable values used in the bind variables in the query statement.</td>
</tr>
<tr>
<td>p_max_row_count</td>
<td>Maximum number of rows returned from the query in p_query which should be added to the collection.</td>
</tr>
<tr>
<td>p_truncate_if_exists</td>
<td>If YES, then members of the collection will first be truncated if the collection exists and no error will be raised. If NO (or not YES), and the collection exists, an error will be raised.</td>
</tr>
</tbody>
</table>

Example

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

```
declare
  l_query varchar2(4000);
Begin
  l_query := 'select empno, sal, comm, deptno, null, hiredate, null, null, null, ename, job, mgr from emp where deptno = :b1';
  APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
    p_collection_name => 'EMPLOYEES',
    p_query => l_query,
    p_names => apex_util.string_to_table('b1'),
    p_values => apex_util.string_to_table('10'));
End;
```
Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the CREATE_COLLECTION_FROM_QUERY_2 method. The query is parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised. It is identical to the CREATE_COLLECTION_FROM_QUERY_B, however, the first five columns of the SELECT clause must be numeric and the next five columns must be date. After the date columns, there can be up to 50 character columns in the SELECT clause.

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

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

Syntax

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

Table 7-14  CREATE_COLLECTION_FROM_QUERYB2 Procedure (No bind version) Parameters

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

Example

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

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

See Also:

- "GET_MEMBER_MD5 Function"
- "CREATE_COLLECTION Procedure"
- "CREATE_OR_TRUNCATE_COLLECTION Procedure"
- "CREATE_COLLECTION_FROM_QUERY Procedure"
- "CREATE_COLLECTION_FROM_QUERY2 Procedure"
- "CREATE_COLLECTION_FROM_QUERY_B Procedure"
7.36 DELETE_ALL_COLLECTIONS Procedure

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

Syntax

APEX_COLLECTION.DELETE_ALL_COLLECTIONS;

Parameters

None.

Example

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

Begin
APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
End;

See Also:

- "DELETE_ALL_COLLECTIONS Procedure,"
- "DELETE_COLLECTION Procedure"
- "DELETE_MEMBER Procedure"
- "DELETE_MEMBERS Procedure"

7.37 DELETE_ALL_COLLECTIONS_SESSION Procedure

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

Syntax

APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;

Parameters

None.
Example

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

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

See Also:

- "DELETE_ALL_COLLECTIONS Procedure"
- "DELETE_COLLECTION Procedure"
- "DELETE_MEMBER Procedure"
- "DELETE_MEMBERS Procedure"

### 7.38 DELETE_COLLECTION Procedure

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to remove all members from and drop. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
</tbody>
</table>

Example

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

```sql
Begin
    APEX_COLLECTION.DELETE_COLLECTION('EMPLOYEE');
End;
```
7.39 DELETE_MEMBER Procedure

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

Syntax

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

Parameters

Table 7-16  DELETE_MEMBER Parameters

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

Example

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

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

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

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

Syntax

\begin{verbatim}
APEX_COLLECTION.DELETE_MEMBERS (  
    p_collection_name IN VARCHAR2,  
    p_attr_number IN VARCHAR2,  
    p_attr_value IN VARCHAR2);
\end{verbatim}

Parameters

\begin{table}[h]  
\centering  
\begin{tabular}{|l|p{10cm}|}  
\hline  
Parameter & Description  \\
\hline  
p\_collection\_name & The name of the collection to delete the specified members from. The maximum length is 255 characters. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist for the current user in the same session.  
\hline  
p\_attr\_number & Attribute number of the member attribute used to match for the specified attribute value for deletion. Valid values are 1 through 50 and null.  
\hline  
p\_attr\_value & Attribute value of the member attribute used to match for deletion. Maximum length can be 4,000 bytes. The attribute value is truncated to 4,000 bytes if greater than this amount.  
\hline  
\end{tabular}  
\caption{DELETE_MEMBERS Parameters}  
\end{table}

Example

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

\begin{verbatim}
Begin  
apex_collection.delete_members(
\end{verbatim}
7.41 GET_MEMBER_MD5 Function

Use this function to compute and return the message digest of the attributes for the member specified by the sequence ID. This computation of message digest is equal to the computation performed natively by collections. Thus, the result of this function could be compared to the MD5_ORIGINAL column of the view apex_collections.

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to add this array of members to. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_seq</td>
<td>Sequence ID of the collection member.</td>
</tr>
</tbody>
</table>
Example
The following example computes the MD5 for the 5th member of the GROCERIES collection.

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

See Also:
- "COLLECTION_HAS_CHANGED Function"
- "RESET_COLLECTION_CHANGED Procedure"
- "RESET_COLLECTION_CHANGED_ALL Procedure"

7.42 MERGE_MEMBERS Procedure

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

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

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

**Syntax**

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.</td>
</tr>
<tr>
<td>p_c001 through p_c050</td>
<td>Array of attribute values to be merged. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. The count of the p_c001 array is used across all arrays. If no values are provided then no actions are performed.</td>
</tr>
<tr>
<td>p_c0xx</td>
<td>Attribute of NN attributes values to be merged. Maximum length can be 4,000 bytes. The attribute value is truncated to 4,000 bytes if greater than this amount.</td>
</tr>
<tr>
<td>p_seq</td>
<td>Identifies the sequence number of the collection to be merged.</td>
</tr>
<tr>
<td>p_null_index</td>
<td>That is if the element identified by this value is null, then treat this row as a null row. For example, if p_null_index is 3, then p_c003 is treated as a null row. In other words, tell the merge function to ignore this row. This results in the null rows being removed from the collection. The null index works with the null value. If the value of the p_cXXX argument is equal to the p_null_value then the row is treated as null.</td>
</tr>
<tr>
<td>p_null_value</td>
<td>Used with the p_null_index argument. Identifies the null value. If used, this value must not be null. A typical value for this argument is &quot;0&quot;</td>
</tr>
<tr>
<td>p_init_query</td>
<td>If the collection does not exist, the collection is created using this query.</td>
</tr>
</tbody>
</table>

Example

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

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

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

7.43 MOVE_MEMBER_DOWN Procedure

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

Syntax

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

Parameters

Table 7-20 MOVE_MEMBER_DOWN Parameters

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

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

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

See Also:

"MOVE_MEMBER_UP Procedure"

7.44 MOVE_MEMBER_UP Procedure

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

Syntax

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

Parameters

Table 7-21 MOVE_MEMBER_UP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist with the specified name of the current user in the same session.</td>
</tr>
<tr>
<td>p_seq</td>
<td>Identifies the sequence number of the collection member to be moved up by one.</td>
</tr>
</tbody>
</table>
Example
This example shows how to a member of the EMPLOYEES collection down one position. After executing this example, sequence ID '5' becomes sequence ID '6' and sequence ID '6' becomes sequence ID '5'.

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

See Also:
"MOVE_MEMBER_DOWN Procedure"

7.45 RESEQUENCE_COLLECTION Procedure

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

Syntax

APEX_COLLECTION.RESEQUENCE_COLLECTION (p_collection_name IN VARCHAR2);

Parameters

Table 7-22  RESEQUENCE_COLLECTION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to resequence. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
</tbody>
</table>

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

BEGIN;
    APEX_COLLECTION.RESEQUENCE_COLLECTION (p_collection_name => 'DEPARTMENTS');
END;
7.46 RESET_COLLECTION_CHANGED Procedure

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

Syntax

APEX_COLLECTION.RESET_COLLECTION_CHANGED ( p_collection_name IN VARCHAR2);

Parameters

Table 7-23  RESET_COLLECTION_CHANGED Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to reset the collection changed flag. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

"RESET_COLLECTION_CHANGED_ALL Procedure"

7.47 RESET_COLLECTION_CHANGED_ALL Procedure

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

APEX_COLLECTION.RESET_COLLECTION_CHANGED_ALL; ( 

Parameters

None.

Example

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

BEGIN;
   APEX_COLLECTION.RESET_COLLECTION_CHANGED_ALL;
END;

See Also:

"RESET_COLLECTION_CHANGED Procedure"

7.48 SORT_MEMBERS Procedure

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

Syntax

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

Parameters

Table 7-24  SORT_MEMBERS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to sort. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_sort_on_column_number</td>
<td>The column number used to sort the collection. The domain of possible values is 1 to 50.</td>
</tr>
</tbody>
</table>
Example

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

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

7.49 TRUNCATE_COLLECTION Procedure

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

Syntax

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

Parameters

Table 7-25    TRUNCATE_COLLECTION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to truncate. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

"CREATE_OR_TRUNCATE_COLLECTION Procedure"
7.50 UPDATE_MEMBER Procedure

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

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

Syntax

APEX_COLLECTION.UPDATE_MEMBER (  
  p_collection_name IN VARCHAR2,  
  p_seq  IN VARCHAR2 DEFAULT NULL,  
  p_c001 IN VARCHAR2 DEFAULT NULL,  
  p_c002 IN VARCHAR2 DEFAULT NULL,  
  p_c003 IN VARCHAR2 DEFAULT NULL,  
  ...  
  p_c050 IN VARCHAR DEFAULT NULL,  
  p_n001 IN NUMBER DEFAULT NULL,  
  p_n002 IN NUMBER DEFAULT NULL,  
  p_n003 IN NUMBER DEFAULT NULL,  
  p_n004 IN NUMBER DEFAULT NULL,  
  p_n005 IN NUMBER DEFAULT NULL,  
  p_d001 IN DATE DEFAULT NULL,  
  p_d002 IN DATE DEFAULT NULL,  
  p_d003 IN DATE DEFAULT NULL,  
  p_d004 IN DATE DEFAULT NULL,  
  p_d005 IN DATE DEFAULT NULL,  
  p_clob001 IN CLOB DEFAULT empty_clob(),  
  p_blob001 IN BLOB DEFAULT empty-blob(),  
  p_xmltype001 IN XMLTYPE DEFAULT NULL);  

Parameters

Note:
Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.
Table 7-26  UPDATE_MEMBER Parameters

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

Example

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

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

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1"
- "UPDATE_MEMBERS Procedure"

7.51 UPDATE_MEMBERS Procedure

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

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

Parameters

Note:

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

Table 7-27  UPDATE_MEMBERS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to update. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.</td>
</tr>
<tr>
<td>p_seq</td>
<td>Array of member sequence IDs to be updated. The count of the p_seq array is used across all arrays.</td>
</tr>
<tr>
<td>p_c001 through p_c050</td>
<td>Array of attribute values to be updated.</td>
</tr>
<tr>
<td>p_n001 through p_n005</td>
<td>Attribute value of numeric</td>
</tr>
<tr>
<td>p_d001 through p_d005</td>
<td>Array of date attribute values to be updated.</td>
</tr>
</tbody>
</table>

Example

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

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

END;

See Also:

"UPDATE_MEMBER Procedure"

7.52 UPDATE_MEMBER_ATTRIBUTE Procedure Signature

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

Syntax

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

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

Example

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

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

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6"
7.53 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2

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

Syntax

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

Parameters

*p_collection_name*

The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

*p_seq*

Sequence ID of the collection member to be updated.

*p_clob_number*

Attribute number of the CLOB member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.

*p_clob_value*

Attribute value of the CLOB member attribute to be updated.

Note:

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

Example

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

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

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

Syntax

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

Parameters

Note:

Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.
Table 7-30  UPDATE_MEMBER_ATTRIBUTE Signature 3 Parameters

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

**Example**

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

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

**See Also:**

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6"

### 7.55 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4

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

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

Parameters

Note:

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

Table 7-31  UPDATE_MEMBER_ATTRIBUTE Signature 4 Parameters

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

Example

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

BEGIN  
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (  
        p_collection_name => 'Departments',  
        p_seq => 2,  
        p_xmltype_number => 1,  
        p_xmltype_value => l_xmltype_content);  
END;
Update the specified NUMBER member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for NUMBER), an error is raised.

**Syntax**

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

**Parameters**

**Note:**

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

**Table 7-32   UPDATE_MEMBER_ATTRIBUTE Signature 5 Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name is converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.</td>
</tr>
<tr>
<td>p_seq</td>
<td>Sequence ID of the collection member to be updated.</td>
</tr>
</tbody>
</table>
Table 7-32  (Cont.) UPDATE_MEMBER_ATTRIBUTE Signature 5 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_attr_number</td>
<td>Attribute number of the NUMBER member attribute to be updated. Valid value is 1 through 5. Any number outside of this range is ignored.</td>
</tr>
<tr>
<td>p_number_value</td>
<td>Attribute value of the NUMBER member attribute to be updated.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6"

7.57 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6

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

Syntax

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

Parameters

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

Table 7-33  UPDATE_MEMBER_ATTRIBUTE Signature 6 Parameters

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

Example

Update the first date attribute of the second collection member in collection named 'Departments', and set it to the value of sysdate.

BEGIN
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (  
    p_collection_name => 'Departments',
    p_seq => 2,
    p_attr_number => 1,
    p_date_value => sysdate );
END;
See Also:

- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4"
- "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5"
APEX_CREDENTIAL

You can use the APEX_CREDENTIAL package to change stored credentials either persistently or for the current APEX session only.

- CLEAR_TOKENS Procedure
- SET_PERSISTENT_CREDENTIALS Procedure Signature 1
- SET_PERSISTENT_CREDENTIALS Procedure Signature 2
- SET_PERSISTENT_TOKEN Procedure
- SET_SESSION_CREDENTIALS Procedure Signature 1
- SET_SESSION_CREDENTIALS Procedure Signature 2
- SET_SESSION_TOKEN Procedure

8.1 CLEAR_TOKENS Procedure

This procedure clears all acquired tokens for a given credential. Applies only to OAuth2 based flows, where the Client ID and Client Secret are used to obtain an Access Token with a certain expiry time. This call clears the obtained tokens.

Syntax

```sql
procedure clear_tokens( p_credential_static_id in varchar2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
</tbody>
</table>

Example

The following example clears all obtained tokens for the credential OAuth Login.

```sql
begin
    apex_credential.clear_tokens(
        p_credential_static_id => 'OAuth Login'
    );
end;
```
8.2 SET_PERSISTENT_CREDENTIALS Procedure Signature 1

This procedure sets Client ID and Client Secret for a given credential. Typically used for the OAuth2 Client Credentials flow. The new credentials are stored persistently and are valid for all current and future sessions.

Syntax

```sql
procedure set_persistent_credentials(
    p_credential_static_id  in varchar2,
    p_client_id             in varchar2,
    p_client_secret         in varchar2 );
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
<tr>
<td>p_client_id</td>
<td>The OAuth2 Client ID.</td>
</tr>
<tr>
<td>p_client_secret</td>
<td>The OAuth2 Client Secret</td>
</tr>
</tbody>
</table>

Example

The following example sets credential OAuth Login.

```sql
begin
    apex_credential.set_persistent_credentials (;
        p_credential_static_id => 'OAuth Login',
        p_client_id => 'dnkjgq237o8832ndj98098-..',
        p_client_secret => '1278672tjksaGSDA789312..');
end;
```

8.3 SET_PERSISTENT_CREDENTIALS Procedure Signature 2

This procedure sets username and password for a given credential. This is typically be used by a security person after application import, and allows to separate responsibilities between a person importing the application and another person storing the credentials.

Syntax

```sql
procedure set_persistent_credentials(
    p_credential_static_id in varchar2,
```
Table 8-3  SET_PERSISTENT_CREDENTIALS Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
<tr>
<td>p_username</td>
<td>The credential username.</td>
</tr>
<tr>
<td>p_password</td>
<td>The credential password.</td>
</tr>
</tbody>
</table>

Example

The following example sets credential Login.

```sql
begin
    apex_credential.set_persistent_credentials (
        p_credential_static_id => 'Login',
        p_username             => 'scott',
        p_password             => 'tiger');
end;
```

8.4 SET_PERSISTENT_TOKEN Procedure

This procedure uses an autonomous transaction in order to store the token in the database table.

Stores a token into a credential store which is obtained with manual or custom PL/SQL code. The credential store saves this token in encrypted form for subsequent use by APEX components. The token is stored for the lifetime of the APEX session. Other sessions cannot use this token. When tokens are obtained with custom PL/SQL code, Client ID, and Client Secret are not stored in that credential store – it contains the tokens set by this procedure only.

Syntax

```sql
procedure set_persistent_token(
    p_credential_static_id in varchar2,
    p_token_type           in t_token_type,
    p_token_value          in varchar2,
    p_token_expires        in date );
```
### Parameters

#### Table 8-4  SET_PERSISTENT_TOKEN Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
<tr>
<td>p_token_type</td>
<td>The token type: APEX_CREDENTIAL.C_TOKEN_ACCESS, APEX_CREDENTIAL.C_TOKEN_REFRESH or APEX_CREDENTIAL.C_TOKEN_ID.</td>
</tr>
<tr>
<td>p_token_value</td>
<td>The value of the token.</td>
</tr>
<tr>
<td>p_token_expiry</td>
<td>The expiry date of the token.</td>
</tr>
</tbody>
</table>

#### Example

The following example stores OAuth2 access token with value sdakjjkhw7632178jh12hs876e38.. and expiry date of 2017-10-31 into credential OAuth Login.

```sql
begin
    apex_credential.set_session_token (  
        p_credential_static_id => 'OAuth Login',  
        p_token_type           => apex_credential.C_TOKEN_ACCESS,  
        p_token_value          => 'sdakjjkhw7632178jh12hs876e38..',  
        p_token_expiry         => to_date('2017-10-31', 'YYYY-MM-DD')  
    );
end;
```

### 8.5 SET_SESSION_CREDENTIALS Procedure Signature1

This procedure sets username and password for a given credential for the current session. Typically used for BASIC authentication when the credentials to be used are to be provided by the end user.

#### Syntax

```sql
procedure set_session_credentials(  
    p_credential_static_id in varchar2,  
    p_username              in varchar2,  
    p_password              in varchar2 );
```

#### Parameters

#### Table 8-5  SET_SESSION_CREDENTIALS Procedure Signature1 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
<tr>
<td>p_username</td>
<td>The credential username.</td>
</tr>
<tr>
<td>p_password</td>
<td>The credential password.</td>
</tr>
</tbody>
</table>
Example
The following example sets credential Login.

begin
    apex_credential.set_session_credentials (p_credential_static_id => 'Login', p_username => 'scott', p_password => 'tiger');
end;

8.6 SET_SESSION_CREDENTIALS Procedure Signature2

This procedure sets Client ID and Client Secret for a given credential for the current session. Typically used for the OAuth2 Client Credentials flow.

Syntax

procedure set_session_credentials(p_credential_static_id  in varchar2, p_client_id             in varchar2, p_client_secret         in varchar2 );

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
<tr>
<td>p_client_id</td>
<td>The OAuth2 Client ID.</td>
</tr>
<tr>
<td>p_client_secret</td>
<td>The OAuth2 Client Secret.</td>
</tr>
</tbody>
</table>

Example

The following example sets credential OAuth Login.

begin
    apex_credential.set_session_credentials (p_credential_static_id => 'OAuth Login', p_client_id => 'dnkjq237o8832ndj98098--..', p_client_secret => '1278672tjksaGSDA789312..');
end;

8.7 SET_SESSION_TOKEN Procedure

This procedure uses an autonomous transaction in order to store the token in the database table.
Stores a token into a credential store which is obtained with manual or custom PL/SQL code. The credential store saves this token in encrypted form for subsequent use by APEX components. The token is stored for the lifetime of the APEX session. Other sessions cannot use this token. When tokens are obtained with custom PL/SQL code, Client ID, and Client Secret are not stored in that credential store – it contains the tokens set by this procedure only.

Syntax

```sql
procedure set_session_token(
    p_credential_static_id  in varchar2,
    p_token_type            in t_token_type,
    p_token_value           in varchar2,
    p_token_expires         in date );
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_credential_static_id</td>
<td>The credential static ID.</td>
</tr>
<tr>
<td>p_token_type</td>
<td>The token type: APEX_CREDENTIAL.C_TOKEN_ACCESS, APEX_CREDENTIAL.C_TOKEN_REFRESH or APEX_CREDENTIAL.C_TOKEN_ID.</td>
</tr>
<tr>
<td>p_token_value</td>
<td>The value of the token.</td>
</tr>
<tr>
<td>p_token_expiry</td>
<td>The expiry date of the token.</td>
</tr>
</tbody>
</table>

Example

The following example stores OAuth2 access token with value sdakjinkhw7632178jhl12hs876e38.. and expiry date of 2017-10-31 into credential OAuth Login.

```sql
begin
    apex_credential.set_session_token (
        p_credential_static_id => 'OAuth Login',
        p_token_type           => apex_credential.C_TOKEN_ACCESS,
        p_token_value          => 'sdakjinkhw7632178jhl12hs876e38..',
        p_token_expiry         => to_date('2017-10-31', 'YYYY-MM-DD') );
end;
```
APEX_CSS

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

- ADD Procedure
- ADD_3RD_PARTY_LIBRARY_FILE Procedure
- ADD_FILE Procedure

9.1 ADD Procedure

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

Syntax

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

Parameters

**Table 9-1 ADD Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_css</td>
<td>The CSS style snippet. For example, #test {color:#fff}</td>
</tr>
<tr>
<td>p_key</td>
<td>Identifier for the style snippet. If specified and a style snippet with the same name has already been added the new style snippet will be ignored.</td>
</tr>
</tbody>
</table>

Example

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

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

9.2 ADD_3RD_PARTY_LIBRARY_FILE Procedure

This procedure adds the link tag to load a 3rd party css file and also takes into account the specified Content Delivery Network for the application. Supported libraries include: jQuery, jQueryUI, jQueryMobile.
If a library has already been added, it is not added a second time.

**Syntax**

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

**Parameters**

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

**Example**

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

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

### 9.3 ADD_FILE Procedure

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

**Syntax**

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

Table 9-3  ADD_FILE Parameters

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

Example

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

apex_css.add_file (  
  p_name => 'jquery.autocomplete',  
  p_directory => p_plugin.image_prefix  );
10

APEX_CUSTOM_AUTH

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

- APPLICATION_PAGE_ITEM_EXISTS Function
- CURRENT_PAGE_IS_PUBLIC Function
- DEFINE_USER_SESSION Procedure
- GET_COOKIE_PROPS Procedure
- GET_LDAP_PROPS Procedure
- GET_NEXT_SESSION_ID Function
- GET_SECURITY_GROUP_ID Function
- GET_SESSION_ID Function
- GET_SESSION_ID_FROM_COOKIE Function
- GET_USER Function
- GET_USERNAME Function
- IS_SESSION_VALID Function
- LOGIN Procedure
- LOGOUT Procedure [DEPRECATED]
- POST_LOGIN Procedure
- SESSION_ID_EXISTS Function
- SET_SESSION_ID Procedure
- SET_SESSION_ID_TO_NEXT_VALUE Procedure
- SET_USER Procedure

10.1 APPLICATION_PAGE_ITEM_EXISTS Function

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

Syntax

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

Table 10-1   APPLICATION_PAGE_ITEM_EXISTS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item_name</td>
<td>The name of the page-level item.</td>
</tr>
</tbody>
</table>

Example

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

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

10.2 CURRENT_PAGE_IS_PUBLIC Function

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

Syntax

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

Example

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

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

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

Syntax

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

Parameters

Table 10-2  DEFINE_USER_SESSION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user</td>
<td>Login name of the user.</td>
</tr>
<tr>
<td>p_session_id</td>
<td>The session ID.</td>
</tr>
</tbody>
</table>

Example

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

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

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

Parameters

Table 10-3  GET_COOKIE_PROPS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>An application ID in the current workspace.</td>
</tr>
<tr>
<td>p_cookie_name</td>
<td>The cookie name.</td>
</tr>
<tr>
<td>p_cookie_path</td>
<td>The cookie path.</td>
</tr>
<tr>
<td>p_cookie_domain</td>
<td>The cookie domain.</td>
</tr>
<tr>
<td>p_secure</td>
<td>Flag to set secure property of cookie.</td>
</tr>
</tbody>
</table>

Example

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

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

10.5 GET_LDAP_PROPS Procedure

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

Syntax

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

Parameters

Table 10-4  GET_LDAP_PROPS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_ldap_host</td>
<td>LDAP host name.</td>
</tr>
<tr>
<td>p_ldap_port</td>
<td>LDAP port number.</td>
</tr>
<tr>
<td>p_use_ssl</td>
<td>Whether SSL is used.</td>
</tr>
<tr>
<td>p_use_exact_dn</td>
<td>Whether exact distinguished names are used.</td>
</tr>
<tr>
<td>p_search_filter</td>
<td>The search filter used if exact DN is not used.</td>
</tr>
<tr>
<td>p_ldap_dn</td>
<td>LDAP DN string.</td>
</tr>
<tr>
<td>p_ldap_edit_function</td>
<td>LDAP edit function name.</td>
</tr>
</tbody>
</table>

Example

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

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

10.6 GET_NEXT_SESSION_ID Function

This function generates the next session ID from the Oracle Application Express sequence generator. This function returns a number.
Syntax
APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID
RETURN NUMBER;

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

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

10.7 GET_SECURITY_GROUP_ID Function

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

Syntax
APEX_CUSTOM_AUTH.GET_SECURITY_GROUP_ID
RETURN NUMBER;

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

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

10.8 GET_SESSION_ID Function

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

Syntax
APEX_CUSTOM_AUTH.GET_SESSION_ID
RETURN NUMBER;
Example
The following example retrieves the session ID for the current user.

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

10.9 GET_SESSION_ID_FROM_COOKIE Function

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

Syntax

APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE
RETURN NUMBER;

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

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

10.10 GET_USER Function

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

Syntax

APEX_CUSTOM_AUTH.GET_USER
RETURN VARCHAR2;

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

DECLARE
  VAL VARCHAR2(256);
BEGIN
  VAL := APEX_CUSTOM_AUTH.GET_USER;
END;
10.11 GET_USERNAME Function

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

Syntax

APEX_CUSTOM_AUTH.GET_USERNAME
RETURN VARCHAR2;

Example

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

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

10.12 IS_SESSION_VALID Function

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

Syntax

APEX_CUSTOM_AUTH.IS_SESSION_VALID
RETURN BOOLEAN;

Example

The following example verifies whether the current session is valid.

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

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

Syntax

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

Parameter

Table 10-5  LOGIN Parameters

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

Example

The following example performs the user authentication and session registration.

```
BEGIN
    APEX_CUSTOM_AUTH.LOGIN (  
        p_uname => 'FRANK', 
        p_password => 'secret99', 
        p_session_id => V('APP_SESSION'), 
        p_app_page => :APP_ID || ':1');
END;
```

Note:

Do not use bind variable notations for p_session_id argument.
10.14 LOGOUT Procedure [DEPRECATED]

**Note:**
This procedure is deprecated. Use `APEX_AUTHENTICATION.LOGOUT` instead.

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

**Syntax**

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

**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_this_app</td>
<td>Current application ID.</td>
</tr>
<tr>
<td>p_next_app_page_sess</td>
<td>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).</td>
</tr>
<tr>
<td>p_next_url</td>
<td>URL to redirect to (use this instead of p_next_app_page_sess).</td>
</tr>
</tbody>
</table>

**Example**

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

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

10.15 POST_LOGIN Procedure

This procedure performs session registration, assuming the authentication step has been completed. It can be called only from within an Oracle Application Express application page context.
APEX_CUSTOM_AUTH.POST_LOGIN(
    p_uname                    IN  VARCHAR2  DEFAULT NULL,
    p_session_id               IN  VARCHAR2  DEFAULT NULL,
    p_app_page                 IN  VARCHAR2  DEFAULT NULL,
    p_preserve_case            IN  BOOLEAN   DEFAULT FALSE);

Table 10-7  POST_LOGIN Parameters

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

Example

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

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

10.16 SESSION_ID_EXISTS Function

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

Syntax

APEX_CUSTOM_AUTH.SESSION_ID_EXISTS
RETURN BOOLEAN;

Example

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

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

### 10.17 SET_SESSION_ID Procedure

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

**Syntax**

APEX_CUSTOM_AUTH.SET_SESSION_ID(  
p_session_id IN NUMBER);

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_session_id</td>
<td>The session ID to be registered.</td>
</tr>
</tbody>
</table>

**Example**

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

APEX_CUSTOM_AUTH.SET_SESSION_ID(APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE);

### 10.18 SET_SESSION_ID_TO_NEXT_VALUE Procedure

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

**Syntax**

APEX_CUSTOM_AUTH.SET_SESSION_ID_TO_NEXT_VALUE;
Example

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

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

10.19 SET_USER Procedure

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user</td>
<td>The user ID to be registered.</td>
</tr>
</tbody>
</table>

Example

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

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

This package contains the implementation for the file parser in APEX. APEX_DATA_PARSER supports XML, JSON, CSV and XLSX files. The most important function in this package is the PARSE function, which is implemented as a table function returning rows of the WWV_FLOW_T_PARSER_ROW type. The parser supports up to 300 columns.

- Global Constants
- Data Types
- DISCOVER Function
- GET_COLUMNS Function
- GET_FILE_PROFILE Function
- GET_FILE_TYPE Function
- GET_XLSX_WORKSHEETS Function
- JSON_TOPROFILE Function
- PARSE Function

11.1 Global Constants

The following constants are used in APEX_DATA_PARSER package.

subtype t_file_type is pls_integer range 1..4;
c_file_type_xlsx constant t_file_type := 1;
c_file_type_csv constant t_file_type := 2;
c_file_type_xml constant t_file_type := 3;
c_file_type_json constant t_file_type := 4;

11.2 Data Types

The data types used by the APEX_DATA_PARSER package are described in this section.

Generic

type t_file_profile is record(
    file_type t_file_type,
    file_charset varchar2(128),
    row_selector varchar2(32767),
    is_single_row boolean,
    first_row_headings boolean,
    xlsx_worksheet varchar2(128),
    xml_namespaces varchar2(4000),
    csv_delimiter varchar2(4),
)
The t_file_columns type is defined as a table of t_file_column type.

type t_file_column is record(
    col_seq       pls_integer,
    name          varchar2(128),
    data_type     apex_exec_api.t_data_type,
    data_type_len pls_integer,
    selector      varchar2(32767),
    decimal_char  varchar2(1),
    group_char    varchar2(1),
    format_mask   varchar2(128) );

11.3 DISCOVER Function

This is a function to discover the column profile of a file. This function calls parse() and then returns the generated file profile. This function is a shortcut which can be used instead of first calling parse() and then get_file_profile().

Syntax

function discover(
    p_content             in blob,
    p_file_name           in varchar2,
    p_decimal_char        in varchar2
default null,
    p_xlsx_sheet_name     in varchar2
default null,
    p_row_selector        in varchar2
default null,
    p_csv_row_delimiter   in varchar2
default LF,
    p_csv_col_delimiter   in varchar2
default null,
    p_csv_enclosed        in varchar2
default '\',
    p_file_charset        in varchar2
default 'AL32UTF8',
    p_max_rows            in number
default 200 ) return clob;

Parameter

Table 11-1 DISCOVER Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_content</td>
<td>The file content to be parsed as a BLOB</td>
</tr>
</tbody>
</table>
Table 11-1  (Cont.) DISCOVER Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_FILE_NAME</td>
<td>The name of the file used to derive the file type.</td>
</tr>
<tr>
<td>P_DECIMAL_CHAR</td>
<td>Use this decimal character when trying to detect NUMBER data types. If not specified, the procedure will auto-detect the decimal character.</td>
</tr>
<tr>
<td>P_XLSX_SHEET_NAME</td>
<td>For XLSX workbooks. The name of the worksheet to parse. If omitted, the function uses the first worksheet found.</td>
</tr>
<tr>
<td>P_ROW_SELECTOR</td>
<td>Whether to detect data types (NUMBER, DATE, TIMESTAMP) during parsing. If set to 'Y', the function will compute the file profile and also add data type information to it. If set to 'N', no data types will be detected and all columns will be VARCHAR2. Default is 'Y'.</td>
</tr>
<tr>
<td>P_DECIMAL_CHAR</td>
<td>Use this decimal character when trying to detect NUMBER data types. If not specified, the procedure will auto-detect the decimal character.</td>
</tr>
<tr>
<td>P_XLSX_SHEET_NAME</td>
<td>For XLSX workbooks. The name of the worksheet to parse. If omitted, the function uses the first worksheet found.</td>
</tr>
<tr>
<td>P_ROW_SELECTOR</td>
<td>For JSON and XML files. Pointer to the array / list of rows within the JSON or XML file. If omitted, the function will: • For XML files: Use &quot;/*&quot; (first tag under the root tag) as the row selector. • For JSON files: Look for a JSON array and use the first array found.</td>
</tr>
<tr>
<td>P_CSV_ROW_DELIMITER</td>
<td>Override the default row delimiter for CSV parsing.</td>
</tr>
<tr>
<td>P_CSV_COL_DELIMITER</td>
<td>Use a specific CSV column delimiter. If omitted, the function detects the column delimiter based on the first row contents.</td>
</tr>
<tr>
<td>P_CSV_ENCLOSED</td>
<td>Override the default enclosure character for CSV parsing.</td>
</tr>
<tr>
<td>P_FILE_CHARSET</td>
<td>File encoding, if not UTF-8 (AL32UTF8).</td>
</tr>
<tr>
<td>P_MAX_ROWS</td>
<td>Stop discovery after P_MAX_ROWS rows have been processed.</td>
</tr>
</tbody>
</table>

Returns

Returns a CLOB containing the file profile in JSON format.

Example

```sql
select wwv_flow_data_parser.discover(
    p_content => (BLOB containing XLS file),
    p_file_name=>'large.xlsx' ) as profile_json
from dual;
```

```
PROFILE_JSON
---------------------------------------------
|
"file-encoding" : "AL32UTF8",
```
11.4 GET_COLUMNS Function

This function returns the columns of a parser profile as a table in order to be consumed by APEX components.

**Syntax**

```
function get_columns( 
p_profile             in clob ) return APEX_T_PARSER_COLUMNS;
```
Parameter

Table 11-2 GET_COLUMNS Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_FILE_PROFILE</td>
<td>File profile to be used for parsing. The file profile might have been computed in a previous PARSE() or DISCOVER() invocation.</td>
</tr>
</tbody>
</table>

Returns

Returns Profile column information as rows of APEX_T_PARSER_COLUMNS.

Example

This example uses DISCOVER() to compute a file profile and then GET_COLUMNS() to return the list of columns among with their data types.

```sql
select *
from table(
  apex_data_parser.get_columns(
    apex_data_parser.discover(
      p_content => {BLOB containing XLSX file},
      p_file_name=>'large.xlsx')));
```

<table>
<thead>
<tr>
<th>COLUMN_POSITION</th>
<th>COLUMN_NAME</th>
<th>DATA_TYPE</th>
<th>FORMAT_MASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C0</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FIRST_NAME</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>LAST_NAME</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GENDER</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>COUNTRY</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>AGE</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>DATE_</td>
<td>DATE</td>
<td>DD&quot;/&quot;MM&quot;/&quot;YYYY</td>
</tr>
<tr>
<td>8</td>
<td>ID</td>
<td>NUMBER</td>
<td></td>
</tr>
</tbody>
</table>

11.5 GET_FILE_PROFILE Function

This function returns the current file profile in JSON format. A file profile is generated when the parse() table function runs and no file profile is passed in. The file profile contains meta data about the parsed files such as the CSV delimiter, the XLSX worksheet name and the columns found during parsing and their data types.

The typical call sequence is as follows:

1. Invoke PARSE - Use this table function to parse the files and get rows and columns in order to display a data preview. While the function runs it computes the file parser profile which can be used in subsequent calls in order to further process the data.

2. Invoke GET_FILE_PROFILE - Retrieve file profile information in JSON format.

3. Process the data
Syntax

```sql
function get_file_profile return clob;
```

Parameter

None.

Returns

Returns file profile of the last `PARSE()` invocation in JSON format.

Example

```sql
select line_number,
       col001,col002,col003,col004,col005,col006,col007,col008
from table(wwv_flow_data_parser.parse(  
    p_content => (BLOB containing XLSX file),
    p_file_name => 'test.xlsx',
    p_xlsx_sheet_name => 'sheet1.xml') );
```

```
<table>
<thead>
<tr>
<th>LINE_NUMBER</th>
<th>COL001</th>
<th>COL002</th>
<th>COL003</th>
<th>COL004</th>
<th>COL005</th>
<th>COL006</th>
<th>COL007</th>
<th>COL008</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>First Name</td>
<td>Last Name</td>
<td>Gender</td>
<td>Country</td>
<td>Age</td>
<td>Date</td>
<td>Id</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Dulce</td>
<td>Abril</td>
<td>Female</td>
<td>United States</td>
<td>32</td>
<td>15/10/2017</td>
<td>1562</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Mara</td>
<td>Hashimoto</td>
<td>Female</td>
<td>Great Britain</td>
<td>25</td>
<td>16/08/2016</td>
<td>1582</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Philip</td>
<td>Gent</td>
<td>Male</td>
<td>France</td>
<td>36</td>
<td>21/05/2015</td>
<td>2587</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Kathleen</td>
<td>Hanner</td>
<td>Female</td>
<td>United States</td>
<td>25</td>
<td>15/10/2017</td>
<td>3549</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Nereida</td>
<td>Magwood</td>
<td>Female</td>
<td>United States</td>
<td>58</td>
<td>16/08/2016</td>
<td>2468</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>Gaston</td>
<td>Brumm</td>
<td>Male</td>
<td>United States</td>
<td>24</td>
<td>21/05/2015</td>
<td>2554</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Etta</td>
<td>Hurn</td>
<td>Female</td>
<td>Great Britain</td>
<td>56</td>
<td>15/10/2017</td>
<td>3598</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td>27</td>
<td>16/08/2016</td>
<td>2456</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>Vincenza</td>
<td>Weiland</td>
<td>Female</td>
<td>United States</td>
<td>40</td>
<td>21/05/2015</td>
<td>6548</td>
</tr>
</tbody>
</table>

select wwv_flow_data_parser.get_file_profile from dual;
```
Chapter 11
GET_FILE_PROFILE Function

```
"xlsx-worksheet" : "sheet1.xml",
"headings-in-first-row" : true,
"file-encoding" : "AL32UTF8",
"single-row" : false,
"parsed-rows" : 2378,
"columns" : [
  {
    "format-mask" : ",",
    "name" : "C0",
    "data-type" : 2,
    "selector" : ""
  },
  {
    "name" : "FIRST_NAME",
    "data-type" : 1,
    "selector" : "",
    "format-mask" : ""
  },
  {
    "selector" : "",
    "data-type" : 1,
    "name" : "LAST_NAME",
    "format-mask" : ""
  },
  {
    "format-mask" : "",
    "data-type" : 1,
    "name" : "GENDER",
    "selector" : ""
  },
  {
    "name" : "COUNTRY",
    "data-type" : 1,
    "selector" : "",
    "format-mask" : ""
  },
  {
    "data-type" : 2,
    "name" : "AGE",
    "selector" : "",
    "format-mask" : ""
  },
  {
    "format-mask" : "DD\"/\"MM\"/\"YYYY",
    "selector" : "",
    "data-type" : 3,
    "name" : "DATE_"
  },
  {
    "name" : "ID",
    "data-type" : 2,
    "selector" : "",
    "format-mask" : ""
  }
],
```
11.6 GET_FILE_TYPE Function

This function returns a file type, based on a file name extension.

Syntax

```sql
function get_file_type( 
   p_file_name in varchar2 ) return t_file_type;
```

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_file_name</td>
<td>File name to get the file type.</td>
</tr>
</tbody>
</table>

Returns

Returns the file type as t_file_type.

Example

```sql
declare
   l_file_type apex_data_parser.t_file_type;
begin
   l_file_type := apex_data_parser.get_file_type( 'test.xlsx' );
end;
```

11.7 GET_XLSX_WORKSHEETS Function

This function returns information on worksheets within an XLSX workbook as a list of apex_t_parser_worksheet instances.

Syntax

```sql
function get_xlsx_worksheets( 
   p_content   in blob  ) return apex_t_parser_worksheets;
```

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_content</td>
<td>XLSX worksheet as a BLOB</td>
</tr>
</tbody>
</table>
Returns
Returns table with worksheet information.

Example

```sql
select * from table(
    apex_data_parser.get_xlsx_worksheets(
        p_content => {BLOB containing XLSX file}
    )
)
```

<table>
<thead>
<tr>
<th>SHEET_SEQUENCE</th>
<th>SHEET_DISPLAY_NAME</th>
<th>SHEET_FILE_NAME</th>
<th>SHEET_PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sheet1</td>
<td>sheet1.xml</td>
<td>worksheets/sheet1.xml</td>
</tr>
</tbody>
</table>

### 11.8 JSON_TO_PROFILE Function

This function converts a file profile in JSON format to an instance of the `t_file_profile` record type.

**Syntax**

```sql
function json_to_profile( p_json inclob ) return t_file_profile;
```

**Parameter**

**Table 11-5 JSON_TO_PROFILE Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_json</td>
<td>The data profile in JSON format.</td>
</tr>
</tbody>
</table>

**Returns**

Returns the the file profile in JSON format.

**Example**

```sql
declare
    l_profile t_file_profile;
begin
    l_profile := apex_data_parser.json_to_profile( '{"file-type", "csv-delimiter" : ",", ... }' );
end;
```
11.9 PARSE Function

This is the main parser function. It allows to parse XML, XLSX, CSV or JSON files and returns a generic table of the following structure:

```
LINE_NUMBER COL001 COL002 COL003 COL004 ... COL300
```

All values are returned in VARCHAR2 format. A returned table row can have a maximum of 300 columns. The maximum length for a table column is 4000 bytes; there is no line length maximum.

File parsing happens on-the-fly as this function is invoked. No data is written to a collection or to a temporary table.

If the `P_FILE_PROFILE` parameter is not passed, the function will compute a file profile with column information during parse. If `P_DETECT_DATA_TYPES` is passed as 'Y' (default), the function also detect column data types during parse. The computed file profile can be retrieved using `GET_FILE_PROFILE` after this function is finished.

1. Invoke PARSE - Use this table function to parse the files and get rows and columns in order to display a data preview.
2. Invoke `GET_FILE_PROFILE` - Retrieve file profile information in JSON format.
3. Process the data - Generate a SQL query based on the data profile to perform custom processing.

**Note:**

- JSON parsing is supported on 11.2 and 12.1.0.1 database versions. In this case, the function uses `APEX_JSON` and `XMLTABLE` functions. For performance reasons it's recommended to upgrade the database to at least 12.2 - JSON parsing is faster by magnitudes on these versions.
- XLSX parsing is done by using `APEX_ZIP` to extract individual XML files from the XLSX archive; the actual XLSX parsing is then done by using the `XMLTABLE` SQL function.

**Syntax**

```sql
function parse(
    p_content          in blob,
    p_file_name        in varchar2     default null,
    p_file_type        in t_file_type  default null,
    p_file_profile     in clob         default null,
    p_detect_data_types in varchar2     default 'Y',
    p_decimal_char     in varchar2     default null,
    p_xlsx_sheet_name  in varchar2     default null,
    p_row_selector     in varchar2     default null,
    p_csv_row_delimiter in varchar2     default LF,
    p_csv_col_delimiter in varchar2     default null,
    p_csv_enclosed     in varchar2     default '',
) return clob;
```
```sql
p_skip_rows in pls_integer default 0,
p_add_headers_row in varchar2 default 'N',
p_file_charset in varchar2 default 'AL32UTF8',
p_max_rows in number default null,
p_return_rows in number default null,
p_store_profile_to_collection in varchar2 default null ) return
wwv_flow_t_parser_table pipelined;
```

### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P_CONTENT</strong></td>
<td>The file content to be parsed as a BLOB</td>
</tr>
<tr>
<td><strong>P_FILE_NAME</strong></td>
<td>The name of the file; only used to derive the file type. Either P_FILE_NAME, P_FILE_TYPE or P_FILE_PROFILE must be passed in.</td>
</tr>
<tr>
<td><strong>P_FILE_TYPE</strong></td>
<td>The type of the file to be parsed. Use this to explicitly pass the file type in. Either P_FILE_NAME, P_FILE_TYPE or P_FILE_PROFILE must be passed in.</td>
</tr>
<tr>
<td><strong>P_FILE_PROFILE</strong></td>
<td>File profile to be used for parsing. The file profile might have been computed in a previous PARSE() invocation. If passed in again, the function will skip some profile detection logic and use the passed in profile - in order to improve performance.</td>
</tr>
<tr>
<td><strong>P_DETECT_DATA_TYPES</strong></td>
<td>Whether to detect data types (NUMBER, DATE, TIMESTAMP) during parsing. If set to 'Y', the function will compute the file profile and also add data type information to it. If set to 'N', no data types will be detected and all columns will be VARCHAR2. Default is 'Y'.</td>
</tr>
<tr>
<td><strong>P_DECIMAL_CHAR</strong></td>
<td>Use this decimal character when trying to detect NUMBER data types. If not specified, the procedure will auto-detect the decimal character.</td>
</tr>
<tr>
<td><strong>P_XLSX_SHEET_NAME</strong></td>
<td>For XLSX workbooks. The name of the worksheet to parse. If omitted, the function uses the first worksheet found.</td>
</tr>
<tr>
<td><strong>P_ROW_SELECTOR</strong></td>
<td>For JSON and XML files. Pointer to the array / list of rows within the JSON or XML file. If omitted, the function will:</td>
</tr>
<tr>
<td></td>
<td>• For XML files: Use &quot;/*/&quot; (first tag under the root tag) as the row selector.</td>
</tr>
<tr>
<td></td>
<td>• For JSON files: Look for a JSON array and use the first array found.</td>
</tr>
<tr>
<td><strong>P_CSV_ROW_DELIMITER</strong></td>
<td>Override the default row delimiter for CSV parsing.</td>
</tr>
<tr>
<td><strong>P_CSV_COL_DELIMITER</strong></td>
<td>Use a specific CSV column delimiter. If omitted, the function will detect the column delimiter based on the first row contents.</td>
</tr>
<tr>
<td><strong>P_CSV_ENCLOSED</strong></td>
<td>Override the default enclosure character for CSV parsing.</td>
</tr>
<tr>
<td><strong>P_SKIP_ROWS</strong></td>
<td>Skip the first N rows when parsing.</td>
</tr>
<tr>
<td><strong>P_ADD_HEADERS_ROW</strong></td>
<td>For XML, JSON: Emit the column headers (tag, attr names) as the first row.</td>
</tr>
<tr>
<td><strong>P_FILE_CHARSET</strong></td>
<td>File encoding, if not UTF-8 (AL32UTF8).</td>
</tr>
<tr>
<td><strong>P_MAX_ROWS</strong></td>
<td>Stop parsing after P_MAX_ROWS have been returned.</td>
</tr>
</tbody>
</table>
Table 11-6  (Cont.) PARSE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_RETURN_ROWS</td>
<td>Amount of rows to return. This is useful when the parser shall to parse more rows (for data type detection), than it is supposed to return. When the specified amount of rows have been emitted, the function will continue parsing (and refining the detected data types) until P_MAX_ROWS has been reached, or until the ROWNUM &lt; x clause of the SQL query kicks in and stops execution.</td>
</tr>
<tr>
<td>P_STORE_PROFILE_TO_COLLECTION</td>
<td>Store the File profile which has been computed during parse into a collection. The collection will be cleared, if it exists. Only be used for computed profiles.</td>
</tr>
</tbody>
</table>

Returns

Returns rows of the APEX_T_PARSER_ROW type.

```
LINE_NUMBER COL001 COL002 COL003 COL004 ... COL300
```

Example

```
select line_number,
       col001, col002, col003, col004, col005, col006, col007, col008
from table(
    wwv_flow_data_parser.parse(
        p_content         => (BLOB containing XLSX spreadsheet),
        p_file_name       => 'test.xlsx',
        p_xlsx_sheet_name => 'sheet1.xml')
) ;
```

<table>
<thead>
<tr>
<th>LINE_NUMBER</th>
<th>COL001</th>
<th>COL002</th>
<th>COL003</th>
<th>COL004</th>
<th>COL005</th>
<th>COL006</th>
<th>COL007</th>
<th>COL008</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>First Name</td>
<td>Last Name</td>
<td>Gender</td>
<td>Country</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Dulce</td>
<td>Abril</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Mara</td>
<td>Hashimoto</td>
<td>Female</td>
<td>Great Britain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Philip</td>
<td>Gent</td>
<td>Male</td>
<td>France</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Kathleen</td>
<td>Hanner</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Nereida</td>
<td>Magwood</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>Gaston</td>
<td>Brumm</td>
<td>Male</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Etta</td>
<td>Hurn</td>
<td>Female</td>
<td>Great Britain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>21</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>23</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>25</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>26</td>
<td>Earlean</td>
<td>Melgar</td>
<td>Female</td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

11-12
<table>
<thead>
<tr>
<th>LINE_NUMBER</th>
<th>COL001</th>
<th>COL002</th>
<th>COL003</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Feature</td>
<td>1.5</td>
<td>41km E of Cape Yakataga, Alaska</td>
</tr>
<tr>
<td></td>
<td>1536513727239</td>
<td>1536514117117</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Feature</td>
<td>0.21</td>
<td>11km ENE of Aguanga, CA</td>
</tr>
<tr>
<td></td>
<td>1536513299520</td>
<td>1536513521231</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Feature</td>
<td>1.84</td>
<td>5km SSW of Pahala, Hawaii</td>
</tr>
<tr>
<td></td>
<td>1536513262940</td>
<td>1536513459610</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Feature</td>
<td>2.55</td>
<td>9km W of Volcano, Hawaii</td>
</tr>
<tr>
<td></td>
<td>1536513100890</td>
<td>1536513446680</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Feature</td>
<td>1.3</td>
<td>62km ESE of Cape Yakataga, Alaska</td>
</tr>
<tr>
<td></td>
<td>1536512917361</td>
<td>1536513322236</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Feature</td>
<td>1.79</td>
<td>7km SW of Tiptonville, Tennessee</td>
</tr>
<tr>
<td></td>
<td>1536512379690</td>
<td>1536512668010</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Feature</td>
<td>1.9</td>
<td>126km NNW of Arctic Village, Alaska</td>
</tr>
<tr>
<td></td>
<td>1536512346186</td>
<td>1536512846567</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Feature</td>
<td>1.4</td>
<td>105km NW of Arctic Village, Alaska</td>
</tr>
<tr>
<td></td>
<td>1536512140162</td>
<td>1536512846334</td>
<td></td>
</tr>
</tbody>
</table>
The **APEX_DEBUG** package provides utility functions for managing the debug message log. Specifically, this package provides the necessary APIs to instrument and debug PL/SQL code contained within your Application Express application as well as PL/SQL code in database stored procedures and functions. Instrumenting your PL/SQL code makes it much easier to track down bugs and isolate unexpected behavior more quickly.

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

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

For further information, see the individual API descriptions.

---

**Note:**

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

- Constants
- DISABLE Procedure
- DISABLE_DBMS_OUTPUT Procedure
- ENABLE Procedure
- ENTER Procedure
- ENABLE_DBMS_OUTPUT Procedure
- ERROR Procedure
- INFO Procedure
- LOG_DBMS_OUTPUT Procedure
- LOG_LONG_MESSAGE Procedure
- LOG_MESSAGE Procedure [Deprecated]
- LOG_PAGE_SESSION_STATE Procedure
- MESSAGE Procedure
- REMOVE_DEBUG_BY_AGE Procedure
12.1 Constants

The following constants are used by this package.

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

12.2 DISABLE Procedure

This procedure turns off debug messaging.

Syntax

```plaintext
APEX_DEBUG.DISABLE;
```

Parameters

None.
Example
This example shows how you can turn off debug messaging.

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

**See Also:**
"ENABLE Procedure"

## 12.3 DISABLE_DBMS_OUTPUT Procedure

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

### Syntax

```
disable_dbms_output;
```

### Parameters

None.

### Example

See `enable_dbms_output`.

**See Also:**
- "ENABLE_DBMS_OUTPUT Procedure"
- "ENABLE Procedure"
- "DISABLE Procedure"

## 12.4 ENABLE Procedure

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

APEX_DEBUG.ENABLE (  
  p_level IN T_LOG_LEVEL DEFAULT C_LOG_LEVEL_INFO );

Parameters

Table 12-1  ENABLE Procedure Parameters

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

Example

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

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

12.5 ENTER Procedure

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

Syntax

APEX_DEBUG.ENTER (  
  p_routine_name IN VARCHAR2,  
  p_name01 IN VARCHAR2 DEFAULT NULL,  
  p_value01 IN VARCHAR2 DEFAULT NULL,  
  p_name02 IN VARCHAR2 DEFAULT NULL,  
  p_value02 IN VARCHAR2 DEFAULT NULL,  
  p_name03 IN VARCHAR2 DEFAULT NULL,  
  p_value03 IN VARCHAR2 DEFAULT NULL,  
  p_name04 IN VARCHAR2 DEFAULT NULL,  
  p_value04 IN VARCHAR2 DEFAULT NULL,  
  p_name05 IN VARCHAR2 DEFAULT NULL,  
  p_value05 IN VARCHAR2 DEFAULT NULL,  
);
Parameters

Table 12-2  APEX_DEBUG.ENTER Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_routine_name</td>
<td>The name of the procedure or function.</td>
</tr>
<tr>
<td>p_namexx/p_valuexx</td>
<td>The procedure or function parameter name and value.</td>
</tr>
<tr>
<td>p_value_max_length</td>
<td>The p_valuexx values is truncated to this length.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "MESSAGE Procedure"
- "ERROR Procedure"
- "WARN Procedure"
- "TRACE Procedure"
- "INFO Procedure"
12.6 ENABLE_DBMS_OUTPUT Procedure

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_prefix</td>
<td>Prefix for lines that go to <code>dbms_output</code>, default '# APEX</td>
</tr>
</tbody>
</table>

Example

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

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

Output:

```
Debug-ERR| 4  
Debug-WRN| 5  
Debug-INF| 7  
# APEX|INF| 8
```
12.7 ERROR Procedure

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>The debug message. Occurrences of '%s' are replaced by p0 to p19, as in <code>utl_lms.format_message</code> and C's <code>sprintf</code>. Occurrences of '%%' represent the special character '%'. Occurrences of '%&lt;n&gt;' are replaced by p&lt;n&gt;.</td>
</tr>
<tr>
<td>p0 through p9</td>
<td>Substitution strings for '%s' placeholders.</td>
</tr>
<tr>
<td>p_max_length</td>
<td>The p&lt;n&gt; values are truncated to this length.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "DISABLE_DBMS_OUTPUT Procedure"
- "ENABLE Procedure"
- "DISABLE Procedure"
12.8 INFO Procedure

This procedure logs messages at level c_log_level_info.

Syntax

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

Parameters

Table 12-5 APEX_DEBUG.INFO Procedure Parameters

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

Example

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

apex_debug.info('Important: %s', 'fnord');
12.9 LOG_DBMS_OUTPUT Procedure

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

Syntax

APEX_DEBUG.LOG_DBMS_OUTPUT;

Parameters

None.

Example

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

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

See Also:

• "MESSAGE Procedure"
• "ERROR Procedure"
• "WARN Procedure"
• "TRACE Procedure"
• "ENTER Procedure"
12.10 LOG_LONG_MESSAGE Procedure

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

Note:
Instead of this procedure, use "ERROR Procedure," "WARN Procedure," "MESSAGE Procedure," "INFO Procedure," "ENTER Procedure," or "TRACE Procedure."

Syntax

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

Parameters

Table 12-6  APEX_DEBUG.LOG_LONG_MESSAGE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>Log long message with maximum size of 32767 bytes.</td>
</tr>
<tr>
<td>p_enabled</td>
<td>Set to TRUE to always log messages, irrespective of whether debugging is enabled. Set to FALSE to only log messages if debugging is enabled.</td>
</tr>
<tr>
<td>p_level</td>
<td>Identifies the level of the long log message. See &quot;Constants.&quot;</td>
</tr>
</tbody>
</table>

Example

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

DECLARE
  l_msg VARCHAR2(32767) := 'Debug outputs anything up to varchar2 limit';
BEGIN
  APEX_DEBUG.ENABLE (p_level => 2);
  APEX_DEBUG.LOG_LONG_MESSAGE (  
    p_message => l_msg,  
  );
12.11 LOG_MESSAGE Procedure [Deprecated]

This procedure logs a debug message.

Note:
Instead of this procedure, use "ERROR Procedure," "WARN Procedure," "MESSAGE Procedure," "INFO Procedure," "ENTER Procedure," or "TRACE Procedure."

Syntax

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

Parameters

Table 12-7 APEX_DEBUG.LOG_MESSAGE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>The debug message with a maximum length of 1000 bytes.</td>
</tr>
<tr>
<td>p_enabled</td>
<td>Messages are logged when logging is enabled, setting a value of TRUE enables logging.</td>
</tr>
<tr>
<td>p_level</td>
<td>Identifies the level of the log message where 1 is most important and 9 is least important. This is an integer value.</td>
</tr>
</tbody>
</table>

Example

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

DECLARE
    l_value varchar2(100) := 'test value';
BEGIN
    APEX_DEBUG.ENABLE (p_level => 2);
    APEX_DEBUG.LOG_MESSAGE(
        p_message => 'l_value = ' || l_value,
        p_level => 1
    );
END;

See Also:
• "MESSAGE Procedure"
• "ERROR Procedure"
• "WARN Procedure"
• "TRACE Procedure"
• "INFO Procedure"

12.12 LOG_PAGE_SESSION_STATE Procedure

This procedure logs the session's item values.

Syntax

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

Parameters

Table 12-8  APEX_DEBUG.LOG_PAGE_SESSION_STATE Procedure Parameters

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

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

BEGIN
  APEX_DEBUG.ENABLE (p_level => 2);

  APEX_DEBUG.LOG_PAGE_SESSION_STATE (p_level => 1);

END;

12.13 MESSAGE Procedure

This procedure logs a formatted debug message, general version.

Syntax

APEX_DEBUG.MESSAGE ( 
  p_message IN VARCHAR2,
  p0  IN VARCHAR2 DEFAULT NULL,
  p1  IN VARCHAR2 DEFAULT NULL,
  p2  IN VARCHAR2 DEFAULT NULL,
  p3  IN VARCHAR2 DEFAULT NULL,
  p4  IN VARCHAR2 DEFAULT NULL,
  p5  IN VARCHAR2 DEFAULT NULL,
  p6  IN VARCHAR2 DEFAULT NULL,
  p7  IN VARCHAR2 DEFAULT NULL,
  p8  IN VARCHAR2 DEFAULT NULL,
  p9  IN VARCHAR2 DEFAULT NULL,
  p10 IN VARCHAR2 DEFAULT NULL,
  p11 IN VARCHAR2 DEFAULT NULL,
  p12 IN VARCHAR2 DEFAULT NULL,
  p13 IN VARCHAR2 DEFAULT NULL,
  p14 IN VARCHAR2 DEFAULT NULL,
  p15 IN VARCHAR2 DEFAULT NULL,
  p16 IN VARCHAR2 DEFAULT NULL,
  p17 IN VARCHAR2 DEFAULT NULL,
  p18 IN VARCHAR2 DEFAULT NULL,
  p19 IN VARCHAR2 DEFAULT NULL,
  p_max_length IN PLS_INTEGER DEFAULT 1000,
  p_level IN T_LOG_LEVEL DEFAULT C_LOG_LEVEL_INFO,
  p_force IN BOOLEAN DEFAULT FALSE );
Parameters

Table 12-9  APEX_DEBUG.MESSAGE Procedure Parameters

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

Example

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

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

See Also:

- "ERROR Procedure"
- "WARN Procedure"
- "TRACE Procedure"
- "INFO Procedure"
- "ENTER Procedure"

12.14 REMOVE_DEBUG_BY_AGE Procedure

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

Syntax

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

Table 12-10  APEX_DEBUG.REMOVE_DEBUG_BY_AGE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID of the application.</td>
</tr>
<tr>
<td>p_older_than_days</td>
<td>The number of days data can exist in the debug message log before it is deleted.</td>
</tr>
</tbody>
</table>

Example

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

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

12.15 REMOVE_DEBUG_BY_APP Procedure

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

Syntax

APEX_DEBUG.REMOVE_DEBUG_BY_APP (  
    p_application_id IN NUMBER);  

Parameters

Table 12-11  APEX_DEBUG.REMOVE_DEBUG_BY_APP Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID of the application.</td>
</tr>
</tbody>
</table>

Example

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

BEGIN
  APEX_DEBUG.REMOVE_DEBUG_BY_APP(  
    p_application_id => TO_NUMBER(:APP_ID) );
END;
12.16 REMOVE_DEBUG_BY_VIEW Procedure

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

Syntax

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

Parameters

Table 12-12 APEX_DEBUG.REMOVE_DEBUG_BY_VIEW Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID of the application.</td>
</tr>
<tr>
<td>p_view_id</td>
<td>The view ID of the view.</td>
</tr>
</tbody>
</table>

Example

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

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

12.17 REMOVE_SESSION_MESSAGES Procedure

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

Syntax

APEX_DEBUG.REMOVE_SESSION_MESSAGES (  
    p_session IN NUMBER DEFAULT NULL);

Parameters

Table 12-13 APEX_DEBUG.REMOVE_SESSION_MESSAGES Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_session</td>
<td>The session ID. Defaults to your current session.</td>
</tr>
</tbody>
</table>
Example

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

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

12.18 TOCHAR Function

This procedure converts a BOOLEAN to a VARCHAR2.

Syntax

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

Parameters

**Table 12-14   APEX_DEBUG.TOCHAR Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>A BOOLEAN 0 or 1 that is converted to FALSE or TRUE respectively.</td>
</tr>
</tbody>
</table>

Example

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

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

12.19 TRACE Procedure

This procedure logs messages at level c_log_level_app_trace.

Syntax

```
APEX_DEBUG.TRACE ( 
    p_message IN VARCHAR2, 
    p0 IN VARCHAR2 DEFAULT NULL, 
```
Parameters

Table 12-15  APEX_DEBUG.TRACE Procedure Parameters

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

Example

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

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

See Also:

- "MESSAGE Procedure"
- "ERROR Procedure"
- "WARN Procedure"
- "ENTER Procedure"
- "INFO Procedure"

12.20 WARN Procedure

This procedure logs messages at level c_log_level_warn.
Syntax

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

Parameters

Table 12-16  APEX_DEBUG.WARN Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>The debug message. Occurrences of '%s' are replaced by p0 to p19, as in utl_lms.format_message and C's sprintf. Occurrences of '%0' represent the special character '%'. Occurrences of '%&lt;n&gt;' are replaced by p&lt;n&gt;.</td>
</tr>
<tr>
<td>p0 through p9</td>
<td>Substitution strings for '%s' placeholders.</td>
</tr>
<tr>
<td>p_max_length</td>
<td>The p&lt;n&gt; values are truncated to this length.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "MESSAGE Procedure"
- "ERROR Procedure"
- "ENTER Procedure"
- "TRACE Procedure"
- "INFO Procedure"
The APEX_ESCAPE package provides functions for escaping special characters in strings to ensure that the data is suitable for further processing.

- Constants
- HTML Function
- HTML_ATTRIBUTE Function
- HTML_TRUNC Function
- HTML_WHITELIST Function
- JS_LITERAL Function
- JSON Function
- LDAP_DN Function
- LDAP_SEARCH_FILTER Function
- NOOP Function
- REGEXP Function
- SET_HTML_ESCAPING_MODE Procedure

### 13.1 Constants

The APEX_ESCAPE package uses the following constants.

```sql
C_LDAP_DN_RESERVED_CHARS CONSTANT VARCHAR2(8) := '"',';','<','>','\';
C_LDAP_SEARCH_RESERVED_CHARS CONSTANT VARCHAR2(5) := '*()\';
C_HTML_WHITELIST_TAGS CONSTANT VARCHAR2(255) := '<h1>,</h1>,<h2>,</h2>,<h3>,</h3>,<h4>,</h4>,<p>,</p>,<b>,</b>,<strong>,</strong>,<i>,</i>,<ul>,</ul>,<ol>,</ol>,<li>,</li>,<br />,<hr />',<hr '/>
```

### 13.2 HTML Function

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

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

The following table, Table 13-1, depicts ascii characters that the function transforms and their escaped values:
Table 13-1  Escaped Values for Transformed ASCII Characters

<table>
<thead>
<tr>
<th>Raw ASCII Characters</th>
<th>Returned Escaped Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>'</td>
<td>'</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

Syntax

APEX_ESCAPE.HTML (  
    p_string IN VARCHAR2 )  
    return VARCHAR2;

Parameters

Table 13-2 describes the parameters available in the HTML function.

Table 13-2  HTML Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The string text that is escaped</td>
</tr>
</tbody>
</table>

Example

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

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

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The text string that is escaped.</td>
</tr>
</tbody>
</table>

Example

See “HTML_TRUNC Function”.

13.4 HTML_TRUNC Function

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

APEX_ESCAPE.HTML_TRUNC (  
    p_string IN CLOB,  
    p_length IN NUMBER DEFAULT 4000 )  
return VARCHAR2;

Parameters

Table 13-4  HTML_TRUNC Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The text string that is escaped.</td>
</tr>
<tr>
<td>p_length</td>
<td>The number of bytes from p_string that are escaped.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "HTML_ATTRIBUTE Function"  
- "HTML Function"  
- "HTML_WHITELIST Function"  
- "SET_HTML_ESCAPING_MODE Procedure"
13.5 HTML_WHITELIST Function

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

Syntax

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

Parameters

Table 13-5  HTML_WHITELIST Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_html</td>
<td>The text string that is filtered.</td>
</tr>
<tr>
<td>p_whitelist_tags</td>
<td>The comma separated list of tags that stays in p_html.</td>
</tr>
</tbody>
</table>

Example

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

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

See Also:

- "HTML_ATTRIBUTE Function"
- "HTML Function"
- "HTML_TRUNC Function"
- "SET_HTML_ESCAPING_MODE Procedure"

13.6 JS_LITERAL Function

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

Syntax

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

Parameters

Table 13-6   JS_LITERAL Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The text string that is escaped.</td>
</tr>
<tr>
<td>p_quote</td>
<td>If not null, this string is placed on the left and right of the result. The quotation character must be a single or a double quotation mark.</td>
</tr>
</tbody>
</table>

Example

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

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

13.7 JSON Function

This function returns p_string with all special characters escaped.

Syntax

APEX_ESCAPE.JSON (  
   p_string  IN VARCHAR2  
)  
RETURN VARCHAR2;

Parameters

Table 13-7   JSON Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The string to be escaped.</td>
</tr>
</tbody>
</table>
Returns/Raised Errors

Table 13-8  JSON Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>The escaped string.</td>
</tr>
</tbody>
</table>

Example

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

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

13.8 LDAP_DN Function

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

Syntax

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

Parameters

Table 13-9  LDAP_DN Function Parameters

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

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

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

Note:

“LDAP_SEARCH_FILTER Function”

13.9 LDAP_SEARCH_FILTER Function

The LDAP_SEARCH_FILTER function escapes reserved characters in an LDAP search filter, according to RFC 4515. The RFC describes *()\ as reserved characters (see p_reserved_chars). These, non-printable characters (ascii 0 - 31) and ones with a code > 127 (see p_escape_non_ascii) are escaped as \xx, where xx is the hexadecimal character code.

Syntax

```
APEX_ESCAPE.LDAP_SEARCH_FILTER ( 
    p_string              IN VARCHAR2,
    p_reserved_chars    IN VARCHAR2 DEFAULT c_ldap_search_reserved_chars,
    p_escape_non_ascii IN BOOLEAN DEFAULT TRUE )
return VARCHAR2;
```

Parameters

Table 13-10 LDAP_SEARCH_FILTER Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The text string that is escaped.</td>
</tr>
<tr>
<td>p_reserved_chars</td>
<td>A list of characters that when found in p_string is escaped with \xx where xx is the character's ASCII hexadecimal code.</td>
</tr>
<tr>
<td>p_escape_non_ascii</td>
<td>If TRUE, characters above ascii 127 in p_string are escaped with \xx where xx is the character's ASCII hexadecimal code. This is supported by RFCs 4514, but may cause errors with older LDAP servers and Microsoft AD.</td>
</tr>
</tbody>
</table>
Example

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

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

Note:

"LDAP_DN Function"

13.10 NOOP Function

Return p_string unchanged. Use this function to silence automatic injection detection tests, similar to dbms_assert.noop for SQL injection.

Syntax

APEX_ESCAPE.NOOP (  
   p_string IN VARCHAR2)  
   return VARCHAR2 deterministic;

Parameters

Table 13-11    APEX_ESCAPE.NOOP Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The input text string.</td>
</tr>
</tbody>
</table>

Example

This example shows how to use NOOP to show the developer's intention to explicitly not escape text.

begin
   sys.htp.p(apex_escape.noop('Cats & Dogs'));
end;
13.11 REGEXP Function

This function escapes characters that can change the context in a regular expression. It should be used to secure user input. The following list depicts ASCII characters that the function escapes with a backslash (\):

\.^$*+-?()[]{}|\n
Syntax

APEX_ESCAPE.REGEXP (  
   p_string IN VARCHAR2);

Parameters

Table 13-12  APEX_ESCAPE.REGEXP Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>Text to escape.</td>
</tr>
</tbody>
</table>

Example

The following example ensures the special character "-" in Mary-Ann will be escaped and ignored by the regular expression engine.

declare  
   l_subscribers varchar2(4000) := 'Christina,Hilary,Mary-Ann,Joel';  
   l_name varchar2(4000) := 'Mary-Ann';  
begin  
   if regexp_instr(l_subscribers,'(^|,)\|\|  
   apex_escape.regexp(l_name)|\|($|,)\|>0  
   then  
      sys.htp.p('found');  
   else  
      sys.htp.p('not found')  
   endif;  
end

13.12 SET_HTML_ESCAPING_MODE Procedure

The SET_HTML_ESCAPING_MODE procedure configures HTML escaping mode for apex_escape.html.

Syntax

APEX_ESCAPE.SET_HTML_ESCAPING_MODE (  
   p_mode IN VARCHAR2);
Parameters

Table 13-13  APEX_ESCAPE.SET_HTML_ESCAPING_MODE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_mode</td>
<td>If equal to B, then do basic escaping, like <code>sys.htf.escape_sc</code>. If equal to E, then do extended escaping.</td>
</tr>
</tbody>
</table>

Example

For an example, see “HTML Function”.

See Also:

- "HTML_WHITELIST Function"
- "HTML Function"
- "HTML_TRUNC Function"
- "HTML_ATTRIBUTE Function"
The APEX_ERROR package provides the interface declarations and some utility functions for an error handling function and includes procedures and functions to raise errors in an Application Express application.

- Constants and Attributes used for Result Types
- Example of an Error Handling Function
- ADD_ERROR Procedure Signature 1
- ADD_ERROR Procedure Signature 2
- ADD_ERROR Procedure Signature 3
- ADD_ERROR Procedure Signature 4
- ADD_ERROR Procedure Signature 5
- APEX_ERROR.HAVE_ERRORS_OCCURRED Function
- AUTO_SET_ASSOCIATED_ITEM Procedure
- EXTRACT_CONSTRAINT_NAME Function
- GET_FIRST_ORA_ERROR_TEXT Function
- INIT_ERROR_RESULT Function

### 14.1 Constants and Attributes used for Result Types

The following constants are used for the API parameter `p_display_location` and the attribute `display_location` in the `t_error` and `t_error_result` types.

```plaintext
c_inline_with_field            constant varchar2(40):='INLINE_WITH_FIELD';
c_inline_with_field_and_notif  constant varchar2(40):='INLINE_WITH_FIELD_AND_NOTIFICATION';
c_inline_in_notification      constant varchar2(40):='INLINE_IN_NOTIFICATION';
c_on_error_page               constant varchar2(40):='ON_ERROR_PAGE';
```

The following constants are used for the API parameter `associated_type` in the `t_error` type.

```plaintext
C_ass_type_page_item           constant varchar2(30):='PAGE_ITEM';
c_ass_type_region              constant varchar2(30):='REGION';
c_ass_type_region_column       constant varchar2(30):='REGION_COLUMN';
```
The following record structure is passed into an error handling callout function and contains all the relevant information about the error.

type t_error is record (  
  message  varchar2(32767), /* Error message which will be displayed */  
  additional_info  varchar2(32767), /* Only used for display_location ON_ERROR_PAGE to display additional error information */  
  display_location  varchar2(40), /* Use constants "used for display_location" below */  
  association_type  varchar2(40), /* Use constants "used for association_type" below */  
  page_item_name  varchar2(255), /* Associated page item name */  
  region_id  number, /* Associated tabular form region id of the primary application */  
  column_alias  varchar2(255), /* Associated tabular form column */  
  row_num  pls_integer, /* Associated tabular form row */  
  is_internal_error  boolean, /* Set to TRUE if it's a critical error raised by the APEX engine, like an invalid SQL/PLSQL statements, ... Internal Errors are always displayed on the Error Page */  
  apex_error_code  varchar2(255), /* Contains the system message code if it's an error raised by APEX */  
  ora_sqlcode  number, /* SQLCODE on exception stack which triggered the error, NULL if the error was not raised by an ORA error */  
  ora_sqlerrm  varchar2(32767), /* SQLERRM which triggered the error, NULL if the error was not raised by an ORA error */  
  error_backtrace  varchar2(32767), /* Output of sys.dbms_utility.format_error_backtrace or sys.dbms_utility.format_call_stack */  
  error_statement  varchar2(32767), /* Statement that was parsed when the error occurred - only suitable when parsing caused the error */  
  component  apex.t_component /* Component which has been processed when the error occurred */  );

The following record structure must be returned by an error handling callout function.

type t_error_result is record (  
  message  varchar2(32767), /* Error message which will be displayed */  
  additional_info  varchar2(32767), /* Only used for display_location ON_ERROR_PAGE to display additional error information */  
  display_location  varchar2(40), /* Use constants "used for display_location" below */  
  page_item_name  varchar2(255), /* Associated page item name */  
  column_alias  varchar2(255) /* Associated tabular form column */  );
14.2 Example of an Error Handling Function

The following is an example of an error handling function.

```sql
create or replace function apex_error_handling_example (p_error in apex_error.t_error )
return apex_error.t_error_result
is
    l_result          apex_error.t_error_result;
    l_reference_id    number;
    l_constraint_name varchar2(255);
begin
    l_result := apex_error.init_error_result (p_error => p_error);

    -- If it's an internal error raised by APEX, like an invalid statement
    -- or code which can't be executed, the error text might contain security
    -- sensitive information. To avoid this security problem we can rewrite the
    -- error to a generic error message and log the original error message for
    -- further investigation by the help desk.
    if p_error.is_internal_error then
        -- mask all errors that are not common runtime errors (Access Denied
        -- errors raised by application / page authorization and all errors
        -- regarding session and session state)
        if not p_error.is_common_runtime_error then
            -- log error for example with an autonomous transaction and
            return
            -- l_reference_id as reference#
            -- l_reference_id := log_error (p_error => p_error);
            --
            -- Change the message to the generic error message which
            doesn't expose
            -- any sensitive information.
            l_result.message         := 'An unexpected internal application error has occurred. Please get in contact with XXX and provide reference# ' ||
            to_char(l_reference_id, '999G999G999G990') || ' for further investigation.';
            l_result.additional_info := null;
        end if;
    else
        -- Always show the error as inline error
        -- Note: If you have created manual tabular forms (using the
```
-- apex_item/htmlldb_item in the SQL statement) you should
still
-- use "On error page" on that pages to avoid loosing
tested data
l_result.display_location := case
when l_result.display_location =
apex_error.c_on_error_page then apex_error.c_inline_in_notification
else l_result.display_location
end;

--
-- Note: If you want to have friendlier ORA error messages, you
can also define
-- a text message with the name pattern APEX.ERROR.ORA-number
-- There is no need to implement custom code for that.
--

-- If it's a constraint violation like
--
-- -) ORA-00001: unique constraint violated
-- -) ORA-02091: transaction rolled back (-> can hide a deferred
constraint)
-- -) ORA-02290: check constraint violated
-- -) ORA-02291: integrity constraint violated - parent key not
found
-- -) ORA-02292: integrity constraint violated - child record
found
--
-- we try to get a friendly error message from our constraint
lookup configuration.
-- If we don't find the constraint in our lookup table we fallback
to
-- the original ORA error message.
if p_error.ora_sqlcode in (-1, -2091, -2290, -2291, -2292) then
    l_constraint_name := apex_error.extract_constraint_name (p_error => p_error);

begin
    select message
    into l_result.message
    from constraint_lookup
    where constraint_name = l_constraint_name;
exception when no_data_found then null; -- not every
constraint has to be in our lookup table
end;
end if;

-- If an ORA error has been raised, for example a
raise_application_error(-20xxx, '...')
-- in a table trigger or in a PL/SQL package called by a process
and we
-- haven't found the error in our lookup table, then we just want
to see
-- the actual error text and not the full error stack with all the
ORA error numbers.
if p_error.ora_sqlcode is not null and l_result.message = p_error.message then
    l_result.message := apex_error.get_first_ora_error_text ( p_error => p_error );
end if;

-- If no associated page item/tabular form column has been set, we can use
-- apex_error.auto_set_associated_item to automatically guess the affected
-- error field by examine the ORA error for constraint names or column names.
if l_result.page_item_name is null and l_result.column_alias is null then
    apex_error.auto_set_associated_item ( p_error => p_error,
                                          p_error_result => l_result );
end if;
end if;

return l_result;
end apex_error_handling_example;

14.3 ADD_ERROR Procedure Signature 1

This procedure adds an error message to the error stack that is used to display an error on an error page or inline in a notification. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of apex_error.c_on_error_page.

Syntax

APEX_ERROR.ADD_ERROR ( p_message          in varchar2,
                         p_additional_info  in varchar2 default null,
                         p_display_location in varchar2 );

Parameters

Table 14-1  ADD_ERROR Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>Displayed error message.</td>
</tr>
</tbody>
</table>
Table 14-1  (Cont.) ADD_ERROR Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_additional_info</td>
<td>Additional error information needed if the error is displayed on the error page.</td>
</tr>
<tr>
<td>p_display_location</td>
<td>Specifies where the error message is displayed. Use the constant apex_error.c_inline_in_notification or apex_error.c_on_error_page. See &quot;Constants and Attributes used for Result Types.&quot;</td>
</tr>
</tbody>
</table>

Example

This example illustrates how to add a custom error message to the error stack. The error message is displayed inline in a notification. This example can be used in a validation or process.

```sql
apex_error.add_error ( p_message          => 'This custom account is not active!',
                       p_display_location => apex_error.c_inline_in_notification );
```

14.4 ADD_ERROR Procedure Signature 2

This procedure adds an error message to the error stack that is used to display an error for a page item inline in a notification. It can be called in a validation or process to add one or more errors to the error stack.

Note:

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of apex_error.c_on_error_page.

Syntax

```sql
APEX_ERROR.ADD_ERROR ( p_message in varchar2,
                        p_additional_info in varchar2 default null,
                        p_display_location in varchar2,
                        p_page_item_name in varchar2 );
```

Parameters

Table 14-2  ADD_ERROR Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>Displayed error message.</td>
</tr>
</tbody>
</table>
Table 14-2  (Cont.) ADD_ERROR Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_additional_info</td>
<td>Additional error information needed if the error is displayed on the error page.</td>
</tr>
<tr>
<td>p_display_location</td>
<td>Specifies where the error message is displayed. Use the constant apex_error.c_inline_with_field or</td>
</tr>
<tr>
<td></td>
<td>apex_error.c_inline_with_field_and_notif. See &quot;Constants and Attributes used for Result Types.&quot;</td>
</tr>
<tr>
<td>p_page_item_name</td>
<td>Name of the page item on the current page that is highlighted if apex_error.c_inline_with_field or</td>
</tr>
<tr>
<td></td>
<td>apex_error.c_inline_with_field_and_notif are used as the display location.</td>
</tr>
</tbody>
</table>

Example

This example illustrates how to add a custom error message to the error stack. The P5_CUSTOMER_ID item is highlighted on the page. The error message is displayed inline in a notification. This example can be used in a validation or process.

```sql
apex_error.add_error (
    p_message           => 'Invalid Customer ID!',
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_page_item_name   => 'P5_CUSTOMER_ID');
```

14.5 ADD_ERROR Procedure Signature 3

This procedure adds an error message to the error stack that is used to display text as defined by a shared component. This error message can be displayed to all display locations. It can be called in a validation or process to add one or more errors to the error stack.

⚠️ **Note:**

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of apex_error.c_on_error_page.

Syntax

```sql
APEX_ERROR.ADD_ERROR (  
    p_error_code          in varchar2,  
    p0                    in varchar2 default null,  
    p1                    in varchar2 default null,  
    p2                    in varchar2 default null,  
    p3                    in varchar2 default null,  
    p4                    in varchar2 default null,  
    p5                    in varchar2 default null,
)
```
Parameters

Table 14-3  ADD_ERROR Procedure Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_error_code</td>
<td>Name of shared component text message.</td>
</tr>
<tr>
<td>p_additional_info</td>
<td>Additional error information needed if the error is displayed on the error page.</td>
</tr>
<tr>
<td>p0 through p9</td>
<td>Values for %0 through %9 placeholders defined in the text message.</td>
</tr>
<tr>
<td>p_escape_placeholders</td>
<td>If set to TRUE, the values provided in p0 through p9 are escaped with sys.htf.escape_sc before replacing the placeholder in the text message. If set to FALSE, values are not escaped.</td>
</tr>
<tr>
<td>p_display_location</td>
<td>Specifies where the error message is displayed. Use the constants defined for p_display_location. See &quot;Constants and Attributes used for Result Types.&quot;</td>
</tr>
<tr>
<td>p_page_item_name</td>
<td>Name of the page item on the current page that is highlighted if apex_error.c_inline_with_field or apex_error.c_inline_with_field_and_notif are used as the display location.</td>
</tr>
</tbody>
</table>

Example

This example illustrates how to add a custom error message, where the text is stored in a text message, to the error stack. The P5_CUSTOMER_ID item is highlighted on the page. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error ( 
    p_error_code => 'INVALID_CUSTOMER_ID',
    p0 => l_customer_id,
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_page_item_name => 'P5_CUSTOMER_ID' );
```

14.6 ADD_ERROR Procedure Signature 4

This procedure adds an error message to the error stack that is used to display an error for a tabular form inline in a notification. It can be called in a validation or process to add one or more errors to the error stack.
**Note:**

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

**Syntax**

```sql
APEX_ERROR.ADD_ERROR (  
p_message          in varchar2,  
p_additional_info  in varchar2 default null,  
p_display_location in varchar2,  
p_region_id        in number,  
p_column_alias     in varchar2 default null,  
p_row_num          in number );
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_message</td>
<td>Displayed error message.</td>
</tr>
<tr>
<td>p_additional_info</td>
<td>Additional error information needed if the error is displayed on the error page.</td>
</tr>
<tr>
<td>p_display_location</td>
<td>Specifies where the error message is displayed. Use the constant <code>apex_error.c_inline_with_field</code> or <code>apex_error.c_inline_with_field_and_notif</code>. See &quot;Constants and Attributes used for Result Types.&quot;</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The ID of a tabular form region on the current page. The ID can be read from the view <code>APEX_APPLICATION_PAGE_REGIONS</code>.</td>
</tr>
<tr>
<td>p_column_alias</td>
<td>Name of a tabular form column alias defined for p_region_id that is highlighted if <code>apex_error.c_inline_with_field</code> or <code>apex_error.c_inline_with_field_and_notif</code> are used as a display location.</td>
</tr>
<tr>
<td>p_row_num</td>
<td>Number of the tabular form row where the error occurred.</td>
</tr>
</tbody>
</table>

**Example**

This example illustrates how to add a custom error message for a tabular form, where the column `CUSTOMER_ID` is highlighted, to the error stack. The error message is displayed inline in a notification. This example can be used in a validation or process.

```sql
APEX_ERROR.ADD_ERROR (  
p_message          => 'Invalid Customer ID!',  
p_display_location => apex_error.c_inline_with_field_and_notif,  
p_region_id        => l_region_id,  
p_column_alias     => 'CUSTOMER_ID',  
p_row_num          => l_row_num );
```
14.7 ADD_ERROR Procedure Signature 5

This procedure adds an error message to the error stack of a tabular form that is used to display text as defined by a shared component. This error message can be displayed to all display locations. It can be called in a validation or process to add one or more errors to the error stack.

**Note:**

This procedure must be called before the Application Express application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

**Syntax**

```sql
APEX_ERROR.ADD_ERROR (  
  p_error_code          in varchar2,  
  p0                    in varchar2 default null,  
  p1                    in varchar2 default null,  
  p2                    in varchar2 default null,  
  p3                    in varchar2 default null,  
  p4                    in varchar2 default null,  
  p5                    in varchar2 default null,  
  p6                    in varchar2 default null,  
  p7                    in varchar2 default null,  
  p8                    in varchar2 default null,  
  p9                    in varchar2 default null,  
  p_escape_placeholders in boolean  default true,  
  p_additional_info     in varchar2 default null,  
  p_display_location    in varchar2,  
  p_region_id           in number,  
  p_column_alias        in varchar2 default null,  
  p_row_num             in number );
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_error_code</td>
<td>Name of shared component text message.</td>
</tr>
<tr>
<td>p0 through p9</td>
<td>Values for %0 through %9 placeholders defined in the text message.</td>
</tr>
<tr>
<td>p_escape_placeholders</td>
<td>If set to TRUE, the values provided in p0 through p9 are escaped with sys.htf.escape_sc before replacing the placeholder in the text message. If set to FALSE, values are not escaped.</td>
</tr>
<tr>
<td>p_additional_info</td>
<td>Additional error information needed if the error is displayed on the error page.</td>
</tr>
</tbody>
</table>

Chapter 14

ADD_ERROR Procedure Signature 5

14-10
Table 14-5  (Cont.) ADD_ERROR Procedure Signature 5 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_display_location</td>
<td>Specifies where the error message is displayed. Use the constants defined for p_display_location. See &quot;Constants and Attributes used for Result Types.&quot;</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The ID of the tabular form region on the current page. The ID can be read from the view APEX_APPLICATION_PAGE_REGIONS.</td>
</tr>
<tr>
<td>p_column_alias</td>
<td>The name of the tabular form column alias defined for p_region_id that is highlighted if apex_error.c_inline_with_field or apex_error.c_inline_with_field_and_notif are used as a display location.</td>
</tr>
<tr>
<td>p_row_num</td>
<td>Number of the tabular form row where the error occurred.</td>
</tr>
</tbody>
</table>

Example

This example illustrates how to add a custom error message, where the text is stored in a text message, to the error stack. The CUSTOMER_ID column on the tabular form is highlighted. The error message is displayed inline in a notification. This example can be used in a validation or process.

```apex
apex_error.add_error (  
  p_error_code       => 'INVALID_CUSTOMER_ID',  
  p0                 => l_customer_id,  
  p_display_location => apex_error.c_inline_with_field_and_notif,  
  p_region_id        => l_region_id,  
  p_column_alias     => 'CUSTOMER_ID',  
  p_row_num          => l_row_num );
```

14.8 APEX_ERROR.HAVE_ERRORS_OCCURRED Function

This function returns TRUE if (inline) errors have occurred and FALSE if no error has occurred.

Syntax

APEX_ERROR.HAVE_ERRORS_OCCURRED return boolean;

Example

This example only executes the statements of the IF statement if no error has been raised.

```apex
if not apex_error.have_errors_occurred then  
  ...  
end if;
```
14.9 AUTO_SET_ASSOCIATED_ITEM Procedure

This procedure automatically sets the associated page item or tabular form column based on a constraint contained in p_error.ora_sqlerrm. This procedure performs the following:

- Identifies the constraint by searching for the schema.constraint pattern.
- Only supports constraints of type P, U, R and C.
- For constraints of type C (check constraints), the procedure parses the expression to identify those columns that are used in the constraints expression.
- Using those columns, the procedure gets the first visible page item or tabular form column that is based on that column and set it as associated p_error_result.page_item_name or p_error_result.column_alias.
- If a page item or tabular form column was found, p_error_result.display_location is set to apex_error.c_inline_with_field_andnotif.

Syntax

APEX_ERROR.AUTO_SET_ASSOCIATED_ITEM (  
    p_error_result in out nocopy t_error_result,  
    p_error        in            t_error );

Parameters

Table 14-6    AUTO_SET_ASSOCIATED_ITEM Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_error_result</td>
<td>The result variable of your error handling function.</td>
</tr>
<tr>
<td>p_error</td>
<td>The p_error parameter of your error handling function.</td>
</tr>
</tbody>
</table>

Example

See an example of how to use this procedure in "Example of an Error Handling Function."

14.10 EXTRACT_CONSTRAINT_NAME Function

This function extracts a constraint name contained in p_error.ora_sqlerrm. The constraint must match the pattern schema.constraint.

Syntax

APEX_ERROR.EXTRACT_CONSTRAINT_NAME (  
    p_error in t_error,  
    p_include_schema in boolean default false )  
return varchar2;
Parameters

Table 14-7  EXTRACT_CONSTRAINT_NAME Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_error</td>
<td>The p_error parameter of your error handling function.</td>
</tr>
<tr>
<td>p_include_schema</td>
<td>If set to TRUE, the result is prefixed with the schema name. For example, HR.DEMO_PRODUCT_INFO_PK. If set to FALSE, only the constraint name is returned.</td>
</tr>
</tbody>
</table>

Example

See an example of how to use this procedure in "Example of an Error Handling Function."

14.11 GET_FIRST_ORA_ERROR_TEXT Function

This function returns the first ORA error message text stored in p_error.ora_sqlerrm. If p_error.ora_sqlerrm does not contain a value, NULL is returned.

Syntax

APEX_ERROR.GET_FIRST_ORA_ERROR_TEXT (  
    p_error    in t_error,  
    p_include_error_no in boolean default false )  
return varchar2;

Parameters

Table 14-8  GET_FIRST_ORA_TEXT Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_error</td>
<td>The p_error parameter of your error handling function.</td>
</tr>
<tr>
<td>p_include_error_no</td>
<td>If set to TRUE, ORA-xxxx is included in the returned error message. If set to FALSE, only the error message text is returned.</td>
</tr>
</tbody>
</table>

Example

See an example of how to use this procedure in "Example of an Error Handling Function."

14.12 INIT_ERROR_RESULT Function

This function returns the t_error_result type initialized with the values stored in p_error.
Note:
This function must be used to ensure initialization is compatible with future changes to t_error_result.

Syntax

APEX_ERROR.INIT_ERROR_RESULT (  
   p_error  in t_error)  
   return t_error_result;

Parameters

Table 14-9  INT_ERROR_RESULT Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_error</td>
<td>The p_error parameter of your error handling function.</td>
</tr>
</tbody>
</table>

Example

See an example of how to use this function in "Example of an Error Handling Function."
15

APEX_EXEC

The APEX_EXEC package encapsulates data processing and querying capabilities and provides an abstraction from the data source to Application Express components and plug-ins. It contains procedures and functions to execute queries or procedural calls on local and remote data sources as well as web source modules. It can be used for plug-in development as well as for procedural PL/SQL processing in applications or within packages and procedures.

All procedures require an APEX session to be set up. In a pure SQL or PL/SQL context, use the APEX_SESSION package to initialize a new session.

The typical call sequence depends on the used data source.

1. REST Enabled SQL Data Sources
   a. Prepare bind variables with [optional]
      • Create a variable of APEX_EXEC.T PARAMETERS type
      • Add bind values with APEX_EXEC.ADD_PARAMETER
   b. Execute the remote query call
      • Call APEX_EXEC.OPEN_REMOTE_SQL_QUERY
   c. Get column indexes for result column names
      • Call APEX_EXEC.OPEN_WEB_SOURCE_QUERY
   d. Loop until the result set is exhausted
      • Call APEX_EXEC.NEXT_ROW
   e. Retrieve column values for each column by position
      • call APEX_EXEC.GET VARCHAR2, APEX_EXEC.GET NUMBER, APEX_EXEC.GET DATE, ...
   f. Finally ALWAYS close the query context - IMPORTANT
      • Call APEX_EXEC.CLOSE

2. Web source module
   a. Prepare web source parameters variables with [optional]
      • Create a variable of APEX_EXEC.T PARAMETERS type
      • Add bind values with APEX_EXEC.ADD_PARAMETER
   b. Prepare filters to be passed to the web source (if supported by the web source)
      • Create a variable of APEX_EXEC.T FILTERS type
      • Add bind values with APEX_EXEC.ADD_FILTER
   c. Prepare order by expressions to be passed to the web source (if supported by the web source)
- Create a variable of `APEX_EXEC.T_ORDER_BYS` type
- Add bind values with `APEX_EXEC.ADD_ORDER_BY`

d. Execute the remote query call
   - Call `APEX_EXEC.OPEN_WEB_SOURCE_QUERY`
   - Pass in filters, order bys and parameters previously prepared

e. Get column indexes for result column names
   - Call `APEX_EXEC.GET_COLUMN_POSITION`

f. Loop until the result set is exhausted
   - Call `APEX_EXEC.NEXT_ROW`

g. Retrieve column values for each column by position
   - Call `APEX_EXEC.GETVARCHAR2, APEX_EXEC.GETNUMBER, APEX_EXEC.GETDATE, ...`

h. Finally ALWAYS close the query context - IMPORTANT
   - Call `APEX_EXEC.CLOSE`

Note:
Always add an exception handler to your procedure or function to ensure that `APEX_EXEC.CLOSE` is being called in any case. This is important to release server resources like database cursors, temporary lobs and others.

- Global Constants
- Data Types
- `ADD_COLUMN` Procedure
- `ADD_DML_ROW` Procedure
- `ADD_FILTER` Procedure
- `ADD_ORDER_BY` Procedure
- `ADD_PARAMETER` Procedure
- `CLEAR_DML_ROWS` Procedure
- `CLOSE` Procedure
- `COPY_DATA` Procedure
- `EXECUTE_DML` Procedure
- `EXECUTE_PLSQL` Procedure
- `EXECUTE_REMOTE_PLSQL` Procedure
- `EXECUTE_WEB_SOURCE` Procedure
- GET Functions
- `GET_COLUMN` Function
- `GET_COLUMN_COUNT` Function
15.1 Global Constants

The following constants are used in APEX_EXEC package.

INTERNAL constants - DO NOT USE

```sql
subtype t_location is varchar2(12);

c_location_local_db constant t_location := 'LOCAL';
c_location_remote_db constant t_location := 'REMOTE';
c_location_web_source constant t_location := 'WEB_SOURCE';

c_lov_shared constant t_lov_type := 1;
c_lov_sql_query constant t_lov_type := 2;
c_lov_static constant t_lov_type := 3;

subtype t_query_type is varchar2(23);

c_query_type_table constant t_query_type := 'TABLE';
```
c_query_type_sql_query constant t_query_type := 'SQL';
c_query_type_func_return_sql constant t_query_type := 'FUNC_BODY_RETURNING_SQL';

subtype t_dml_operation is pls_integer range 1..3;

c_dml_operation_insert constant t_dml_operation := 1;
c_dml_operation_update constant t_dml_operation := 2;
c_dml_operation_delete constant t_dml_operation := 3;

subtype t_target_type is varchar2(13);

c_target_type_region_source constant t_target_type := 'REGION_SOURCE';
c_target_type_table constant t_target_type := 'TABLE';
c_target_type_sql_query constant t_target_type := 'SQL';
c_target_type_plsql constant t_target_type := 'PLSQL_CODE';

subtype t_post_processing is pls_integer range 1..3;

c_postprocess_where_orderby constant t_post_processing := 1;
c_postprocess_sql constant t_post_processing := 2;
c_postprocess_plsql_return_sql constant t_post_processing := 3;

Data type constants to be used in the ADD_FILTER or ADD_COLUMN procedures.

subtype t_data_type is pls_integer range 1..13;

c_data_type_varchar2 constant t_data_type := 1;
c_data_type_number constant t_data_type := 2;
c_data_type_date constant t_data_type := 3;
c_data_type_timestamp constant t_data_type := 4;
c_data_type_timestamp_tz constant t_data_type := 5;
c_data_type_timestamp_ltz constant t_data_type := 6;
c_data_type_interval_y2m constant t_data_type := 7;
c_data_type_interval_d2s constant t_data_type := 8;
c_data_type_blob constant t_data_type := 9;
c_data_type_bfile constant t_data_type := 10;
c_data_type_clob constant t_data_type := 11;
c_data_type_rowid constant t_data_type := 12;
c_data_type_user_defined constant t_data_type := 13;

Filter type constants to be used in the ADD_FILTER procedures.

c_filter_eq constant t_filter_type := 1;
c_filter_not_eq constant t_filter_type := 2;
c_filter_gt constant t_filter_type := 3;
c_filter_gte constant t_filter_type := 4;
c_filter_lt constant t_filter_type := 5;
c_filter_lte constant t_filter_type := 6;
c_filter_null constant t_filter_type := 7;
c_filter_not_null constant t_filter_type := 8;
c_filter_starts_with constant t_filter_type := 9;
c_filter_not_starts_with constant t_filter_type := 10;
c_filter_ends_with constant t_filter_type := 11;
c_filter_not_ends_with constant t_filter_type := 12;
c_filter_contains constant t_filter_type := 13;
c_filter_not_contains  constant t_filter_type := 14;
c_filter_in          constant t_filter_type := 15;
c_filter_not_in      constant t_filter_type := 16;
c_filter_between    constant t_filter_type := 17;
c_filter_not_between constant t_filter_type := 18;
c_filter_regexp     constant t_filter_type := 19;
-- date filters: days/months/...
c_filter_last       constant t_filter_type := 20;
c_filter_not_last   constant t_filter_type := 21;
c_filter_next       constant t_filter_type := 22;
c_filter_not_next   constant t_filter_type := 23;

-- interactive reports
  c_filter_like       constant t_filter_type := 24;
c_filter_not_like   constant t_filter_type := 25;
c_filter_search     constant t_filter_type := 26;
c_filter_sql_expression constant t_filter_type := 27;

  c_filter_expr_sep    constant varchar2(1) := '-';
c_filter_expr_value_sep constant varchar2(1) := chr(1);

  -- interval types for date filters (last, not last, next, not next)
c_filter_int_type_year constant t_filter_interval_type := 'Y';
c_filter_int_type_month constant t_filter_interval_type := 'M';
c_filter_int_type_week  constant t_filter_interval_type := 'W';
c_filter_int_type_day   constant t_filter_interval_type := 'D';
c_filter_int_type_hour  constant t_filter_interval_type := 'H';
c_filter_int_type_minute constant t_filter_interval_type := 'MI';

Order by constants to be used in the ADD_FILTER procedures.

  c_order_asc           constant t_order_direction := 1;
c_order_desc          constant t_order_direction := 2;

  c_order_nulls_first   constant t_order_nulls := 1;
c_order_nulls_last    constant t_order_nulls := 2;

Constants or empty filter, order by, columns or parameter arrays

  c_empty_columns         t_columns;
c_empty_filters         t_filters;
c_empty_order_bys       t_order_bys;
c_empty_parameters      t_parameters;

15.2 Data Types

The data types used by the APEX_EXEC package are described in this section.

Generic

  subtype t_column_name is varchar2(32767); t_column
type t_value is record (
    varchar2_value      varchar2(32767),
    number_value        number,
    date_value          date,
    timestamp_value     timestamp,
    timestamp_tz_value  timestamp with time zone,
    timestamp_ltz_value timestamp with local time zone,
    interval_y2m_value  interval year to month,
    interval_d2s_value  interval day to second,
    blob_value          blob,
    bfile_value         bfile,
    clob_value          clob,
    anydata_value       sys.anydata );

type t_values is table of t_value index by pls_integer;

Bind variables

type t_parameter is record (
    name      t_column_name,
    data_type t_data_type,
    value     t_value );

type t_parameters is table of t_parameter index by pls_integer;

Filters

subtype t_filter_type          is pls_integer range 1..27;
subtype t_filter_interval_type is varchar2(2);

type t_filter is record (
    column_name       t_column_name,
    data_type         t_data_type,
    filter_type       t_filter_type,
    filter_values     t_values,
    sql_expression    varchar2(32767),
    search_columns    t_columns,
    null_result       boolean default false,
    is_case_sensitive boolean default true );

type t_filters is table of t_filter index by pls_integer;

Order Bys

subtype t_order_direction is pls_integer range 1..2;
subtype t_order_nulls     is pls_integer range 1..2;

type t_order_by is record (
    column_name   t_column_name,
    direction     t_order_direction,
    order_nulls   t_order_nulls );

type t_order_bys is table of t_order_by index by pls_integer;
Columns

type t_column is record (
    name                   t_column_name,
    sql_expression         varchar2(4000),
    --
    data_type              t_data_type,
    data_type_length       pls_integer,
    format_mask            varchar2(4000),
    --
    is_required            boolean default false,
    is_primary_key         boolean default false,
    is_query_only          boolean default false,
    is_checksum            boolean default false,
    is_returning           boolean default false );

type t_columns is table of t_column index by pls_integer;

Context Handle

subtype t_context is pls_integer;

Data Source Capabilities

type t_source_capabilities is record(
    location               t_location,
    --
    pagination             boolean default false,
    --
    allow_fetch_all_rows   boolean default false,
    --
    filtering              boolean default false,
    order_by               boolean default false,
    group_by               boolean default false,
    --
    filter_eq              boolean default false,
    filter_not_eq          boolean default false,
    filter_gt              boolean default false,
    filter_gte             boolean default false,
    filter_lt              boolean default false,
    filter_lte             boolean default false,
    filter_null            boolean default false,
    filter_not_null        boolean default false,
    filter_contains        boolean default false,
    filter_not_contains    boolean default false,
    filter_like            boolean default false,
    filter_not_like        boolean default false,
    filter_starts_with     boolean default false,
    filter_not_starts_with boolean default false,
    filter_between         boolean default false,
    filter_not_between     boolean default false,
    filter_in              boolean default false,
    filter_not_in          boolean default false,
15.3 ADD_COLUMN Procedure

This procedure adds a column to the columns collection. Columns collections can be passed the OPEN_*_CONTEXT calls in order to request only a subset of columns. This is particularly useful for web sources where no SQL statement is involved. If no or an empty column array is passed, all columns defined in the web source are being fetched.

Syntax

procedure add_column(
    p_columns         in out nocopy t_columns,
    p_column_name     in            varchar2,
    p_data_type       in            t_data_type default null,
    p_sql_expression  in            varchar2    default null,
    p_format_mask     in            varchar2    default null,
    p_is_primary_key  in            boolean     default false,
    p_is_query_only   in            boolean     default false,
    p_is_returning    in            boolean     default false,
    p_is_checksum     in            boolean     default false );

Parameters

Table 15-1  ADD_COLUMN Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_columns</td>
<td>Columns array.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name.</td>
</tr>
<tr>
<td>p_data_type</td>
<td>Column data type.</td>
</tr>
<tr>
<td>p_sql_expression</td>
<td>SQL expression in order to derive a column from other columns.</td>
</tr>
<tr>
<td>p_format_mask</td>
<td>Format mask to use for this column.</td>
</tr>
<tr>
<td>p_is_primary_key</td>
<td>Whether this is a primary key column (default false).</td>
</tr>
<tr>
<td>p_is_query_only</td>
<td>Query only columns are not written in a DML context (default false).</td>
</tr>
<tr>
<td>p_is_returning</td>
<td>Whether to retrieve the RETURNING column after DML has been executed (default false).</td>
</tr>
</tbody>
</table>
Table 15-1  (Cont.) ADD_COLUMN Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_is_checksum</td>
<td>Whether this is a checksum (row version) column (default false).</td>
</tr>
</tbody>
</table>

Example

```sql
declare
    l_columns     apex_exec.t_columns;
    l_context     apex_exec.t_context;
begin
    apex_exec.add_column(
        p_columns  => l_columns,
        p_column_name => 'ENAME');

    apex_exec.add_column(
        p_columns  => l_columns,
        p_column_name => 'SAL');

    l_context := apex_exec.open_web_source_query(
        p_module_static_id => '{web source module static ID}',
        p_columns          => l_columns,
        p_max_rows         => 1000);

    while apex_exec.next_row( l_context ) loop
        -- process rows here ...
    end loop;

    apex_exec.close( l_context );
exception
    when others then
        apex_exec.close( l_context );
        raise;
end;
```

15.4 ADD_DML_ROW Procedure

This procedure adds one row to the DML context. This is called after the `open_dml_context` and before the `execute_dml` procedures. This procedure can be called multiple times to process multiple rows. All columns of the new row are initialized with NULL.

Use `set_value`, `set_null` and `set_row_version_checksum` to populate the new row with values and the checksum for lost-update detection.

**Syntax**

```sql
procedure add_dml_row(
    p_context               in t_context,
    p_operation             in t_dml_operation );
```
Parameters

Table 15-2  ADD_DML_ROW Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions</td>
</tr>
<tr>
<td>p_operation</td>
<td>DML operation to be executed on this row. Possible values:</td>
</tr>
<tr>
<td></td>
<td>• c_dml_operation_insert</td>
</tr>
<tr>
<td></td>
<td>• c_dml_operation_update</td>
</tr>
<tr>
<td></td>
<td>• c_dml_operation_delete</td>
</tr>
</tbody>
</table>

Example

See "OPEN_REMOTE_DML_CONTEXT Function"
",""OPEN_WEB__SOURCE_DML_CONTEXT Function"
",""OPEN_LOCAL_DML_CONTEXT Function"

15.5 ADD_FILTER Procedure

This procedure adds a filter to the filter collection.

Syntax

```sql
procedure add_filter (  
p_filters           in out nocopy t_filters,  
p_filter_type       in            t_filter_type,  
p_column_name       in            t_column_name );

procedure add_filter (  
p_filters           in out nocopy t_filters,  
p_filter_type       in            t_filter_type,  
p_column_name       in            t_column_name,  
p_value             in            varchar2,  
p_null_result       in            boolean  default false,  
p_is_case_sensitive in            boolean  default true );

procedure add_filter (  
p_filters           in out nocopy t_filters,  
p_filter_type       in            t_filter_type,  
p_column_name       in            t_column_name,  
p_from_value        in            varchar2,  
p_to_value          in            varchar2,  
p_null_result       in            boolean  default false,  
p_is_case_sensitive in            boolean  default true );

procedure add_filter (  
p_filters           in out nocopy t_filters,  
p_filter_type       in            t_filter_type,  
p_column_name       in            t_column_name,  
p_values            in            wwv_flow_t_varchar2,  
p_null_result       in            boolean  default false,  
p_is_case_sensitive in            boolean  default true );
```
procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_value             in            number,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_from_value        in            number,  
  p_to_value          in            number,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_values            in            wwv_flow_t_number,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_value             in            date,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_from_value        in            date,  
  p_to_value          in            date,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_value             in            timestamp,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_from_value        in            timestamp,  
  p_to_value          in            timestamp,  
  p_null_result       in            boolean default false );

procedure add_filter (  
  p_filters           in out nocopy t_filters,  
  p_filter_type       in            t_filter_type,  
  p_column_name       in            t_column_name,  
  p_null_result       in            boolean default false );
procedure add_filter (  
  p_filters in out nocopy t_filters,  
  p_filter_type in t_filter_type,  
  p_column_name in t_column_name,  
  p_from_value in timestamp with time zone,  
  p_to_value in timestamp with time zone,  
  p_null_result in boolean default false );

procedure add_filter (  
  p_filters in out nocopy t_filters,  
  p_filter_type in t_filter_type,  
  p_column_name in t_column_name,  
  p_from_value in timestamp with local time zone,  
  p_to_value in timestamp with local time zone,  
  p_null_result in boolean default false );

procedure add_filter (  
  p_filters in out nocopy t_filters,  
  p_filter_type in t_filter_type,  
  p_column_name in t_column_name,  
  p_from_value in timestamp with local time zone,  
  p_to_value in timestamp with local time zone,  
  p_null_result in boolean default false );

procedure add_filter (  
  p_filters in out nocopy t_filters,  
  p_filter_type in t_filter_type,  
  p_column_name in t_column_name,  
  p_interval in pls_integer,  
  p_interval_type in t_filter_interval_type,  
  p_null_result in boolean default false );

procedure add_filter (  
  p_filters in out nocopy t_filters,  
  p_search_columns in t_columns,  
  p_is_case_sensitive in boolean default false,  
  p_value in varchar2 );

procedure add_filter (  
  p_filters in out nocopy t_filters,  
  p_sql_expression in varchar2 );

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_filters</td>
<td>Filters array.</td>
</tr>
<tr>
<td>p_filter_type</td>
<td>Type of filter - use one of the t_filter_type constants.</td>
</tr>
</tbody>
</table>
### Table 15-3  (Cont.) ADD_FILTER Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_column_name</td>
<td>Column to apply this filter on.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value for filters requiring one value (e.g. equals or greater than).</td>
</tr>
<tr>
<td>p_values</td>
<td>Value array for IN or NOT IN filters.</td>
</tr>
<tr>
<td>p_from_value</td>
<td>Lower value for filters requiring a range (e.g. between).</td>
</tr>
<tr>
<td>p_to_value</td>
<td>Upper value for filters requiring a range (e.g. between).</td>
</tr>
<tr>
<td>p_interval</td>
<td>Interval for date filters (e.g. last X months).</td>
</tr>
<tr>
<td>p_interval_type</td>
<td>Interval type for date filters (months, dates).</td>
</tr>
<tr>
<td>p_sql_expression</td>
<td>Generic SQL expression to use as filter.</td>
</tr>
<tr>
<td>p_null_result</td>
<td>Result to return when the actual column value is NULL.</td>
</tr>
<tr>
<td>p_is_case_sensitive</td>
<td>Whether this filter should work case-sensitive or not.</td>
</tr>
<tr>
<td>p_search_columns</td>
<td>List of columns to apply the row search filter on.</td>
</tr>
</tbody>
</table>

### Example

```sql
declare
  l_filters     apex_exec.t_filters;
  l_context     apex_exec.t_context;
begin
  apex_exec.add_filter(
    p_filters     => l_filters,
    p_filter_type => apex_exec.c_filter_eq,
    p_column_name => 'ENAME',
    p_value       => 'KING');
  apex_exec.add_filter(
    p_filters     => l_filters,
    p_filter_type => apex_exec.c_filter_gt,
    p_column_name => 'SAL',
    p_value       => 2000);
  l_context := apex_exec.open_web_source_query(
    p_module_static_id => '{web source module static ID}',
    p_filters          => l_filters,
    p_max_rows         => 1000);
  while apex_exec.next_row( l_context ) loop
    -- process rows here ...
  end loop;
  apex_exec.close( l_context );
exception
  when others then
    apex_exec.close( l_context );
```

[Chapter 15: ADD_FILTER Procedure]
15.6 ADD_ORDER_BY Procedure

This procedure adds an order by expression to the order bys collection.

Syntax

procedure add_order_by ( 
    p_order_bys   in out nocopy t_order_bys, 
    p_position   in    pls_integer, 
    p_direction  in    t_order_direction default c_order_asc, 
    p_order_nulls in    t_order_nulls     default null );

procedure add_order_by ( 
    p_order_bys   in out nocopy t_order_bys, 
    p_column_name in    t_column_name, 
    p_direction   in    t_order_direction default c_order_asc, 
    p_order_nulls in    t_order_nulls     default null );

Parameters

Table 15-4  ADD_ORDER_BY Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_order_bys</td>
<td>Order by collection.</td>
</tr>
<tr>
<td>p_position</td>
<td>References a column of the provided data source by position.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>References a column name or alias of the provided data source.</td>
</tr>
<tr>
<td>p_direction</td>
<td>Defines if the column should be sorted ascending or descending. Valid values are c_order_asc and c_order_desc.</td>
</tr>
<tr>
<td>p_order_nulls</td>
<td>Defines if NULL data will sort to the bottom or top. Valid values are NULL, c_order_nulls_first and c_order_nulls_last. Use NULL for automatic handling based on the sort direction.</td>
</tr>
</tbody>
</table>

Example

declare 
    l_order_bys   apex_exec.t_order_bys; 
    l_context     apex_exec.t_context; 
begin 
    apex_exec.add_order_by( 
        p_order_bys => l_order_bys, 
        p_column_name => 'ENAME', 
        p_direction  => apex_exec.c_order_asc );
l_context := apex_exec.open_web_source_query(
    p_module_static_id => 'web source module static ID',
    p_order_bys        => l_order_bys,
    p_max_rows         => 1000);

while apex_exec.next_row( l_context ) loop
    -- process rows here ...
    end loop;

    apex_exec.close( l_context );

exception
    when others then
        apex_exec.close( l_context );
        raise;
end;

15.7 ADD_PARAMETER Procedure

This procedure adds a SQL parameter to the parameter collection. To use SQL parameters, prepare the array first, then use it in the execution call.

Syntax

procedure add_parameter (p_parameters in out nocopy t_parameters,
    p_name in t_column_name,
    p_value in varchar2);

procedure add_parameter (p_parameters in out nocopy t_parameters,
    p_name in t_column_name,
    p_value in number);

procedure add_parameter (p_parameters in out nocopy t_parameters,
    p_name in t_column_name,
    p_value in date);

procedure add_parameter (p_parameters in out nocopy t_parameters,
    p_name in t_column_name,
    p_value in timestamp);

procedure add_parameter (p_parameters in out nocopy t_parameters,
    p_name in t_column_name,
    p_value in timestamp with time zone);

procedure add_parameter (p_parameters in out nocopy t_parameters,
    p_name in t_column_name,
    p_value in timestamp with local time zone);
procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            interval year to month );

procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            interval day to second );

procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            blob );

procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            bfile );

procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            clob );

procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            sys.anydata );

procedure add_parameter (  
  p_parameters in out nocopy t_parameters,  
  p_name       in            t_column_name,  
  p_value      in            t_data_type,  
  p_data_type  in            t_column_name,  
  p_value      in            t_value );

Parameters

Table 15-5  ADD_PARAMETER Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_parameters</td>
<td>SQL parameter array.</td>
</tr>
<tr>
<td>p_name</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>p_value</td>
<td>Parameter value.</td>
</tr>
</tbody>
</table>

Example

declare  
  l_parameters  apex_exec.t_parameters;  
begin  
apex_exec.add_parameter( l_parameters, 'ENAME', 'SCOTT' );  
apex_exec.add_parameter( l_parameters, 'SAL', 2000 );
apex_exec.add_parameter( l_parameters, 'HIREDATE', sysdate );

apex_exec.execute_remote_plsql(
    p_server_static_id => '{static ID of the REST Enabled SQL Service}',
    p_auto_bind_items => false,
    p_plsql_code => q'#begin insert into emp values (:ENAME, :SAL, :HIREDATE ); end;#',
    p_sql_parameters => l_parameters );
end;

15.8 CLEAR_DML_ROWS Procedure

This procedure clears all DML rows which have been added with add_dml_rows.

Syntax

procedure clear_dml_rows(
    p_context in t_context );

Parameters

Table 15-6  CLEAR_DML_ROWS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions</td>
</tr>
</tbody>
</table>

15.9 CLOSE Procedure

This procedure closes the query context and releases resources.

Note:

Ensure to always call this procedure after work has finished or an exception occurs.

Syntax

procedure close( 
    p_context in t_context );
15.10 COPY_DATA Procedure

This procedure fetches all rows from the source context and writes to the target context. Useful to copy data between different data sources (for example, local to remote, remote to web source etc).

Syntax

```plsql
procedure copy_data(
    p_from_context          in out nocopy t_context,
    p_to_context            in out nocopy t_context,
    p_operation_column_name in            varchar2 default null);
```

Parameters

Table 15-8    COPY_DATA Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_from_context</td>
<td>Query context to fetch rows from.</td>
</tr>
<tr>
<td>p_to_context</td>
<td>DML context to write rows to.</td>
</tr>
<tr>
<td>p_operation_column_name</td>
<td>Column in the query context to indicate the DML operation to execute on the target context. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• &quot;I&quot;: insert the row on the target (DML) context</td>
</tr>
<tr>
<td></td>
<td>• &quot;U&quot;: update the row on the target (DML) context</td>
</tr>
<tr>
<td></td>
<td>• &quot;D&quot;: delete the row on the target (DML) context</td>
</tr>
</tbody>
</table>

Example

```plsql
declare
    l_columns        apex_exec.t_columns;
    l_dml_context    apex_exec.t_context;
    l_query_context  apex_exec.t_context;
begin
    -- I. Define DML columns
    apex_exec.add_column(
        p_columns          => l_columns,
        p_column_name      => 'EMPNO',
        p_data_type        => apex_exec.c_data_type_number,
        p_is_primary_key   => true );
    apex_exec.add_column(
        p_columns          => l_columns,
        p_column_name      => 'ENAME',
```
p_data_type => apex_exec.c_data_type_varchar2);
apex_exec.add_column(
  p_columns => l_columns,
  p_column_name => 'JOB',
  p_data_type => apex_exec.c_data_type_varchar2);
apex_exec.add_column(
  p_columns => l_columns,
  p_column_name => 'HIREDATE',
  p_data_type => apex_exec.c_data_type_date);
apex_exec.add_column(
  p_columns => l_columns,
  p_column_name => 'MGR',
  p_data_type => apex_exec.c_data_type_number);
apex_exec.add_column(
  p_columns => l_columns,
  p_column_name => 'SAL',
  p_data_type => apex_exec.c_data_type_number);
apex_exec.add_column(
  p_columns => l_columns,
  p_column_name => 'COMM',
  p_data_type => apex_exec.c_data_type_number);
apex_exec.add_column(
  p_columns => l_columns,
  p_column_name => 'DEPTNO',
  p_data_type => apex_exec.c_data_type_number);

-- II. Open the Query Context object
l_query_context := apex_exec.open_remote_sql_query(
  p_server_static_id => 'DevOps_Remote_SQL',
  p_sql_query => 'select * from emp',
  p_columns => l_columns);

-- III. Open the DML context object
l_dml_context := apex_exec.open_remote_dml_context(
  p_server_static_id => '{remote server static id}',
  p_columns => l_columns,
  p_query_type => apex_exec.c_query_type_sql_query,
  p_sql_query => 'select * from emp ');

-- IV. Copy rows
apex_exec.copy_data(
  p_from_context => l_query_context,
  p_to_context => l_dml_context);

-- V. Close contexts and free resources
apex_exec.close( l_dml_context );
apex_exec.close( l_query_context );
exception
  when others then
    apex_exec.close( l_dml_context );
    apex_exec.close( l_query_context );
    raise;
end;
15.11 EXECUTE_DML Procedure

This procedure executes the DML context. This procedure is called after:

- After the context has been opened (open_dml_context).
- One or many DML rows have been added with add_dml_row.
- Column values have been set with set_values, set_null or set_value.

Syntax

```plsql
procedure execute_dml(
    p_context               in t_context,
    p_continue_on_error     in boolean default false );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_continue_on_error</td>
<td>Whether to continue executing DML for the remaining rows after an error occurred (defaults to false).</td>
</tr>
</tbody>
</table>

Example


15.12 EXECUTE_PLSQL Procedure

This procedure executes PL/SQL code based on the current process or plug-in location settings.

Syntax

```plsql
procedure execute_plsql (  
    p_plsql_code in varchar2,  
    p_auto_bind_items in boolean default true,  
    p_sql_parameters in out t_parameters );
```
Parameters

Table 15-10  EXECUTE_PLSQL Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plsql_code</td>
<td>PL/SQL code to be executed. Based on the settings of the current process or process-type plug-in, the code is executed locally or remotely.</td>
</tr>
<tr>
<td>p_auto_bind_items</td>
<td>Whether to automatically bind page item values for IN and OUT direction. If the PL/SQL code references bind variables which are not page items, this must be set to false. Default: true</td>
</tr>
<tr>
<td>p_sql_parameters</td>
<td>Additional bind variables; if needed.</td>
</tr>
</tbody>
</table>

Examples

Example 1
Executes a PL/SQL block with arbitrary bind variables, so any bind can be used to pass values and to get values back.

declare
    l_sql_parameters apex_exec.t_parameters;
    l_out_value      varchar2(32767);
begin
    apex_exec.add_parameter( l_sql_parameters, 'MY_BIND_IN_VAR', '{some value}' );
    apex_exec.add_parameter( l_sql_parameters, 'MY_BIND_OUT_VAR', '' );
    apex_exec.execute_plsql( p_plsql_code => q'begin :
        :MY_BIND_OUT_VAR := some_plsql( p_parameter => :MY_BIND_IN_VAR ); end;' ,
        p_auto_bind_items => false,
        p_sql_parameters  => l_sql_parameters );

    l_out_value := apex_exec.get_parameter_varchar2( p_parameters => l_sql_parameters,
        p_name       => 'MY_BIND_OUT_VAR' );

    -- further processing of l_out_value
end;

Example 2
Executes a PL/SQL block.

begin
    apex_exec.execute_plsql( p_plsql_code => q'begin :
        P10_NEW_SAL :=' }
salary_pkg.raise_sal( p_empno => :P10_EMPNO ); end;#"");
end;

15.13 EXECUTE_REMOTE_PLSQL Procedure

This procedure executes PL/SQL code on a REST Enabled SQL instance.

Syntax

```plsql
procedure execute_remote_plsql(
    p_server_static_id     in     varchar2,
    p_plsql_code           in     varchar2,
    p_auto_bind_items      in     boolean      default true,
    p_sql_parameters       in out t_parameters );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_server_static_id</td>
<td>Static ID of the ORDS REST Enabled SQL Instance.</td>
</tr>
<tr>
<td>p_plsql_code</td>
<td>PL/SQL code to be executed.</td>
</tr>
<tr>
<td>p_auto_bind_items</td>
<td>Whether to automatically bind page item values for IN <em>and</em> OUT direction. If the PL/SQL code references bind variables which are not page items, this must be set to FALSE. Default: TRUE</td>
</tr>
<tr>
<td>p_sql_parameters</td>
<td>Additional bind variables; if needed.</td>
</tr>
</tbody>
</table>

Examples

Example 1

Executes a PL/SQL block on a remote database.

begin
    apex_exec.execute_remote_plsql(
        p_server_static_id => '{Static ID of the REST Enabled SQL Service}',
        p_plsql_code => q'begin :P10_NEW_SAL := salary_pkg.raise_sal( p_empno => :P10_EMPNO ); end;#'");
end;

Example 2

Works with arbitrary bind variables, so any bind can be used to pass values to the REST Enabled SQL service and to get values back.

declare
    l_sql_parameters apex_exec.t_parameters;
    l_out_value      varchar2(32767);
begin
  apex_exec.add_parameter( l_sql_parameters, 'MY_BIND_IN_VAR', '
    (some value)' );
  apex_exec.add_parameter( l_sql_parameters, 'MY_BIND_OUT_VAR', '' );
  apex_exec.execute_remote_plsql( 
    p_server_static_id => 'Static ID of the REST Enabled SQL Service', 
    p_plsql_code => q'#begin :MY_BIND_OUT_VAR := some_remote_plsql( p_parameter => :MY_BIND_IN_VAR ); end;#', 
    p_auto_bind_items => false, 
    p_sql_parameters => l_sql_parameters );
  l_out_value := apex_exec.get_parameter_varchar2( 
    p_parameters => l_sql_parameters, 
    p_name => 'MY_BIND_OUT_VAR');
  -- further processing of l_out_value
end;

15.14 EXECUTE_WEB_SOURCE Procedure

This procedure executes a web source operation based on module name, operation and URL pattern (if required). Use the t_parameters array to pass in values for declared web source parameters. Web Source invocation is done based on metadata defined in Shared Components.

Syntax

procedure execute_web_source ( 
  p_module_static_id in varchar2,
  p_operation        in varchar2,
  p_url_pattern      in varchar2         default null,
  p_parameters       in out t_parameters );

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_module_static_id</td>
<td>Static ID of the web source module.</td>
</tr>
<tr>
<td>p_operation</td>
<td>Name of the operation, e.g. POST, GET, DELETE.</td>
</tr>
<tr>
<td>p_url_pattern</td>
<td>If multiple operations with the same name exist, specify the URL pattern, as defined in Shared Components, to identify the web source operation.</td>
</tr>
<tr>
<td>p_parameters</td>
<td>Parameter values to pass to the external web source.</td>
</tr>
</tbody>
</table>

Note that HTTP Headers, URL Patterns and other parameters being passed to a Web Source Module are typically strings. Oracle recommends to explicitly pass all values to VARCHAR2 before adding to the T_PARAMETERS array.
Table 15-12  (Cont.) EXECUTE_WEB_SOURCE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>n/a</td>
</tr>
<tr>
<td>p_parameters</td>
<td>Array with OUT parameter values, received from the web source module.</td>
</tr>
</tbody>
</table>

Example

This example assumes a REST service being created on the EMP table using ORDS and the “Auto-REST” feature (ORDS.ENABLE_OBJECT). Then a Web Source Module for this REST service is being created in Shared Components as “ORDS EMP”.

The POST operation has the following "Request Body Template" defined:

```
{ "empno": "#EMPNO#", "ename": "#ENAME#", "job": "#JOB#", "sal": "#SAL#" }
```

Parameters are defined as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPNO</td>
<td>IN</td>
<td>Request Body</td>
</tr>
<tr>
<td>ENAME</td>
<td>IN</td>
<td>Request Body</td>
</tr>
<tr>
<td>SAL</td>
<td>IN</td>
<td>Request Body</td>
</tr>
<tr>
<td>JOB</td>
<td>IN</td>
<td>Request Body</td>
</tr>
<tr>
<td>RESPONSE</td>
<td>OUT</td>
<td>Request Body</td>
</tr>
<tr>
<td>Content-Type</td>
<td>IN</td>
<td>HTTP Header</td>
</tr>
</tbody>
</table>

PL/SQL code to invoke that web source operation looks as follows:

```
declare
    l_params apex_exec.t_parameters;
begin
    apex_exec.add_parameter( l_params, 'ENAME', :P2_ENAME );
    apex_exec.add_parameter( l_params, 'EMPNO', :P2_EMPNO );
    apex_exec.add_parameter( l_params, 'SAL', :P2_SAL );
    apex_exec.add_parameter( l_params, 'JOB', :P2_JOB );

    apex_exec.execute_web_source( 
        p_module_static_id => 'ORDS_EMP',
        p_operation        => 'POST',
        p_parameters       => l_params );

    :P2_RESPONSE := apex_exec.get_parameter_clob(l_params,'RESPONSE');
end;
```

15.15 GET Functions

These functions retrieve column values. There is one function for each supported data type.
Syntax

```plaintext
function get_varchar2 (  
  p_context     in t_context,  
  p_column_idx  in pls_integer ) return varchar2;

function get_number (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return number;

function get_date (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return date;

function get_timestamp (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return timestamp;

function get_timestamp_tz (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return timestamp with time zone;

function get_timestamp_ltz (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return timestamp with local time zone;

function get_clob (  
  p_context     in t_context,  
  p_column_idx in pls_integer ) return clob;

function get_intervald2s (  
  p_context     in t_context,  
  p_column_idx in pls_integer ) return interval day to second;

function get_intervaly2m (  
  p_context     in t_context,  
  p_column_idx in pls_integer ) return interval year to month;
```

Parameters

Table 15-13  GET Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_column_idx</td>
<td>Column index.</td>
</tr>
</tbody>
</table>

Returns

Column value as specific data type.
15.16 GET_COLUMN Function

This function returns detailed information about a result set column.

Syntax

```plaintext
function get_column(
    p_context    in t_context,
    p_column_idx in pls_integer ) return t_column;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_functions.</td>
</tr>
<tr>
<td>p_column_idx</td>
<td>Index of the column to retrieve information for.</td>
</tr>
</tbody>
</table>

Returns

`t_column object with column metadata.`

15.17 GET_COLUMN_COUNT Function

This function returns the result column count for the current execution context.

Syntax

```plaintext
function get_column_count ( 
    p_context in t_context ) return pls_integer;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_functions.</td>
</tr>
</tbody>
</table>

Returns

Returns the result columns count.

15.18 GET_COLUMN_POSITION Function

This function returns the array index for a given column alias. In order to do this lookup operation only once, Oracle recommends you to use GET_COLUMN_POSITION function before entering the NEXT_ROW loop. This saves on computing resources.
Syntax

function get_column_position (  
    p_context           in t_context,  
    p_column_name       in varchar2,  
    p_attribute_label   in varchar2  default null,  
    p_is_required       in boolean   default false,  
    p_data_type         in varchar2  default c_data_type_varchar2 )  
return pls_integer;

Parameters

Table 15-16  GET_COLUMN_POSITION Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_attribute_label</td>
<td>Attribute label to format error messages.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name.</td>
</tr>
<tr>
<td>p_is_required</td>
<td>Indicates whether this is a required column.</td>
</tr>
<tr>
<td>p_data_type</td>
<td>Indicates the requested data type.</td>
</tr>
</tbody>
</table>

Returns

The position of the column in the query result set. Throws an exception when p_is_required or p_data_type prerequisites are not met.

15.19 GET_DATA_TYPE Functions

This function converts the t_data_type constant into the VARCHAR2 representation. Converts a data type VARCHAR2 representation to the t_data_type constant.

Syntax

function get_data_type(  
    p_datatype_num in apex_exec.t_data_type ) return varchar2;

Signature 1

function get_data_type(  
    p_datatype_num in varchar2 ) return apex_exec.t_data_type;
15.20 GET Functions

This function retrieves column values for different data types.

Syntax

function get_varchar2 ( 
    p_context     in t_context, 
    p_column_idx  in pls_integer ) return varchar2;

function get_varchar2 ( 
    p_context     in t_context, 
    p_column_name in varchar2 ) return varchar2;

Signature 1

function get_number ( 
    p_context     in t_context, 
    p_column_idx  in pls_integer ) return number;

function get_number ( 
    p_context     in t_context, 
    p_column_name in varchar2 ) return number;

Signature 2

function get_date ( 
    p_context     in t_context, 
    p_column_idx  in pls_integer ) return date;

function get_date ( 
    p_context     in t_context, 
    p_column_name in varchar2 ) return date;
Signature 3

function get_timestamp (  
    p_context in t_context,  
    p_column_idx in pls_integer ) return timestamp;

function get_timestamp (  
    p_context in t_context,  
    p_column_name in varchar2 ) return timestamp;

Signature 4

function get_timestamp_tz (  
    p_context in t_context,  
    p_column_idx in pls_integer ) return timestamp with time zone;

function get_timestamp_tz (  
    p_context in t_context,  
    p_column_name in varchar2 ) return timestamp with time zone;

Signature 5

function get_timestamp_ltz (  
    p_context in t_context,  
    p_column_idx in pls_integer ) return timestamp with local time zone;

function get_timestamp_ltz (  
    p_context in t_context,  
    p_column_name in varchar2 ) return timestamp with local time zone;

Signature 6

function get_clob (  
    p_context in t_context,  
    p_column_idx in pls_integer ) return clob;

function get_clob (  
    p_context in t_context,  
    p_column_name in varchar2 ) return clob;

Signature 7

function get_blob (  
    p_context in t_context,  
    p_column_idx in pls_integer ) return blob;

function get_blob (  
    p_context in t_context,  
    p_column_name in varchar2 ) return blob;
Signature 8

function get_intervald2s (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return dsinterval_unconstrained;

function get_intervald2s (  
  p_context    in t_context,  
  p_column_name in varchar2 ) return dsinterval_unconstrained;

Signature 9

function get_intervaly2m (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return yminterval_unconstrained;

function get_intervaly2m (  
  p_context    in t_context,  
  p_column_name in varchar2 ) return yminterval_unconstrained;

Signature 10

function get_anydata (  
  p_context    in t_context,  
  p_column_idx in pls_integer ) return sys.anydata;

function get_anydata (  
  p_context    in t_context,  
  p_column_name in varchar2 ) return sys.anydata;

Parameters

Table 15-18  GET Functions Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_column_idx</td>
<td>Column index.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name.</td>
</tr>
</tbody>
</table>

Returns

The column value as specific data type

15.21 GET_DML_STATUS_CODE Function

This function returns the SQL status code of the last context execution, for the current row. For local or remote SQL contexts, the ORA error code will be returned in case of an error, NULL in case of success.

For Web Source Module contexts, the function returns the HTTP status code.
Syntax

function get_dml_status_code(
    p_context               in t_context) return number;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_functions.</td>
</tr>
</tbody>
</table>

Returns

The DML status code of the current row.

15.22 GET_DML_STATUS_MESSAGE Function

This function returns the SQL status message of the last context execution, for the current row. For local or remote SQL contexts, the ORA error message will be returned in case of an error, NULL in case of success.

For Web Source Module contexts, the function returns the HTTP reason phrase.

Syntax

function get_dml_status_message(
    p_context               in t_context) return varchar2;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_functions.</td>
</tr>
</tbody>
</table>

Returns

The DML status message of the current row.

15.23 GET_PARAMETER Functions

These functions return a SQL parameter value. Typically used to retrieve values for OUT binds of an executed SQL or PL/SQL statement.
Syntax

function get_parameter_varchar2(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return varchar2;

function get_parameter_number(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return number;

function get_parameter_date(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return date;

function get_parameter_timestamp(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return timestamp;

function get_parameter_timestamp_tz(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return timestamp with time zone;

function get_parameter_timestamp_ltz(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return timestamp with local time zone;

function get_parameter_clob(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return clob;

function get_parameter_interval_d2s(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return interval day to second;

function get_parameter_interval_y2m(
    p_parameters      in t_parameters,
    p_name            in varchar2 ) return interval year to month;

Parameters

Table 15-21  GET_PARAMETER Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_parameters</td>
<td>SQL parameter array.</td>
</tr>
<tr>
<td>p_name</td>
<td>Parameter name.</td>
</tr>
</tbody>
</table>

Returns

Parameter value
15.24 GET_ROW_VERSION_CHECKSUM Function

This function returns the row version checksum for the current row. This is either a specific column (when designated as "checksum column") or a calculated checksum from all column values.

Syntax

```plaintext
function get_row_version_checksum(
    p_context in t_context ) return varchar2;
```

Parameters

Table 15-22  GET_ROW_VERSION_CHECKSUM Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
</tbody>
</table>

Returns

The row version checksum.

15.25 GET_TOTAL_ROW_COUNT Function

This function returns the total row count of the query result.

Syntax

```plaintext
function get_total_row_count ( p_context in t_context ) return pls_integer;
```

Parameters

Table 15-23  GET_TOTAL_ROW_COUNT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
</tbody>
</table>

Returns

The total row count; NULL if unknown.

15.26 HAS_ERROR Function

This function returns the when DML execution led to an error and false, when not.
Syntax

function has_error(
    p_context               in t_context) return boolean;

Parameters

Table 15-24   HAS_ERROR Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_functions.</td>
</tr>
</tbody>
</table>

Returns

true, if an error occurred, false otherwise.

15.27 IS_REMOTE_SQL_AUTH_VALID Function

This function checks whether the current authentication credentials are correct for the given REST Enabled SQL instance.

Syntax

function is_remote_sql_auth_valid(
    p_server_static_id     in     varchar2 ) return boolean;

Parameters

Table 15-25   IS_REMOTE_SQL_AUTH_VALID Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_server_static_id</td>
<td>Static ID of the REST enabled SQL instance.</td>
</tr>
</tbody>
</table>

Returns

Returns true, when credentials are correct, false otherwise.

Example

The following example requires a REST enabled SQL instance created as "My Remote SQL". It uses credentials stored as SCOTT_Credentials.

```sql
begin
    apex_credentials.set_session_credentials(
        p_application_id    => {application-id},
        p_credential_name   => 'SCOTT_Credentials',
        p_username          => 'SCOTT',
        p_password          => '****' );
    if apex_exec.check_rest_enabled_sql_auth(
```
then
  sys.dbms_output.put_line( 'credentials are correct!');
else
  sys.dbms_output.put_line( 'credentials are NOT correct!');
end if;
end;

15.28 NEXT_ROW Function

This function advances the cursor of an open query or DML context, after execution, by one row.

Syntax

function next_row(  
  p_context in t_context ) return boolean;

Parameters

Table 15-26    NEXT_ROW Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
</tbody>
</table>

Returns

Returns false when the end of the response has been reached, true otherwise.

15.29 OPEN_LOCAL_DML_CONTEXT Function

This function opens a DML context based for a local database.

Syntax

function open_local_dml_context (  
  p_columns               in t_columns               default c_empty_columns,
  p_query_type            in t_query_type,            --
  p_table_owner           in varchar2                default null,
  p_table_name            in varchar2                default null,
  p_where_clause          in varchar2                default null,
  --
  p_sql_query             in varchar2                default null,
  p_plsql_function_body   in varchar2                default null,
  --
  p_with_check_option     in boolean                 default true,
  p_optimizer_hint        in varchar2                default null,
  --
  p_dml_table_owner       in varchar2                default null,  
)
 Parameters

**Table 15-27 OPEN_LOCAL_DML_CONTEXT Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_columns</td>
<td>DML columns to pass to the data source.</td>
</tr>
<tr>
<td>p_query_type</td>
<td>DML columns to pass to the data source. Indicates the type of the data source: possible values are:</td>
</tr>
<tr>
<td></td>
<td>• c_query_type_table: Use a plain Table as the data source.</td>
</tr>
<tr>
<td></td>
<td>• c_query_type_sql_query: Use a SQL query as the data source.</td>
</tr>
<tr>
<td></td>
<td>• c_query_type_func_return_sql: Use the SQL query returned by the PL/SQL function.</td>
</tr>
<tr>
<td>p_table_owner</td>
<td>For query type TABLE: Table owner</td>
</tr>
<tr>
<td>p_table_name</td>
<td>For query type TABLE: Table name</td>
</tr>
<tr>
<td>p_where_clause</td>
<td>For query type TABLE: where clause</td>
</tr>
<tr>
<td>p_sql_query</td>
<td>For query type SQL QUERY: the query</td>
</tr>
<tr>
<td>p_plsql_function_body</td>
<td>For query type PLSQL: the PL/SQL function which returns the SQL query</td>
</tr>
<tr>
<td>p_with_check_option</td>
<td>Specify whether to the &quot;WITH CHECK OPTION&quot; should be added to the data source. If set to &quot;true&quot; (default), INSERTED or UPDATED rows cannot violate the where clause.</td>
</tr>
<tr>
<td>p_optimizer_hint</td>
<td>Optimizer hints to be added to the DML clause</td>
</tr>
<tr>
<td>p_dml_table_owner</td>
<td>When set, DML statements will be executed against this table</td>
</tr>
<tr>
<td>p_dml_table_name</td>
<td>When set, DML statements will be executed against this table</td>
</tr>
<tr>
<td>p_dml_plsql_code</td>
<td>Custom PL/SQL code to be executed instead of DML statements</td>
</tr>
<tr>
<td>p_lost_update_detection</td>
<td>lost-update detection type. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• c_lost_update_implicit: APEX calculates a checksum from the row values</td>
</tr>
<tr>
<td></td>
<td>• c_lost_update_explicit: One of the p_columns has the &quot;is_checksum&quot; attribute set</td>
</tr>
<tr>
<td></td>
<td>• c_lost_update_none: No lost update detection</td>
</tr>
</tbody>
</table>
### Table 15-27  (Cont.) OPEN_LOCAL_DML_CONTEXT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lock_rows</td>
<td>Specify whether to lock the rows for the (short) time frame between the lost update detection and the actual DML statement. Possible values:</td>
</tr>
<tr>
<td></td>
<td>• c_lock_rows_automatic: use a SELECT .. FOR UPDATE</td>
</tr>
<tr>
<td></td>
<td>• c_lock_rows_plsql: use custom PL/SQL code to lock the rows</td>
</tr>
<tr>
<td></td>
<td>• c_lock_rows_none: do not lock rows</td>
</tr>
<tr>
<td>p_dml_plsql_code</td>
<td>Custom PL/SQL code to be used to lock the rows</td>
</tr>
<tr>
<td>p_sql_parameters</td>
<td>Bind variables to be used</td>
</tr>
</tbody>
</table>

### Example

The following inserts one row into the EMP table on a REST Enabled SQL Service.

```sql
declare
  l_columns        apex_exec.t_columns;
  l_context        apex_exec.t_context;
begin
  -- I. Define DML columns
  apex_exec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'EMPNO',
    p_data_type    => apex_exec.c_data_type_number,
    p_is_primary_key => true );
  apexExec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'ENAME',
    p_data_type    => apex_exec.c_data_type_varchar2 );
  apexExec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'JOB',
    p_data_type    => apex_exec.c_data_type_varchar2 );
  apexExec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'HIREDATE',
    p_data_type    => apex_exec.c_data_type_date );
  apexExec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'MGR',
    p_data_type    => apex_exec.c_data_type_number );
  apexExec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'SAL',
    p_data_type    => apex_exec.c_data_type_number );
  apexExec.add_column(
    p_columns      => l_columns,
    p_column_name  => 'COMM',
    p_data_type    => apex_exec.c_data_type_number );
```
apex_exec.add_column(
    p_columns => l_columns,
    p_column_name => 'DEPTNO',
    p_data_type => apex_exec.c_data_type_number);

-- II. Open the context object
l_context := apex_exec.open_local_dml_context(
    p_columns => l_columns,
    p_query_type => apex_exec.c_query_type_sql_query,
    p_sql_query => 'select * from emp where deptno = 10',
    p_lost_update_detection => apex_exec.c_lost_update_none);

-- III. Provide DML data
apex_exec.add_dml_row(
    p_context => l_context,
    p_operation => apex_exec.c_dml_operation_insert);

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 1,
    p_value => 4711);

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 2,
    p_value => 'DOE');

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 3,
    p_value => 'DEVELOPR');

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 4,
    p_value => sysdate);

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 6,
    p_value => 1000);

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 8,
    p_value => 10);

-- IV: Execute the DML statement
apex_exec.execute_dml(
    p_context => l_context,
    p_continue_on_error => false);

apex_exec.close( l_context );

exception
    when others then
        apex_exec.close( l_context );
        raise;
end;
Returns
The context object representing the DML handle.

15.30 OPEN_REMOTE_DML_CONTEXT Function
This function opens a DML context based for a remote database.

Syntax

```sql
function open_remote_dml_context (  
    p_server_static_id      in varchar2,  
    --
    p_columns               in t_columns               default c_empty_columns,
    p_query_type            in t_query_type,  
    --
    p_table_owner           in varchar2                default null,
    p_table_name            in varchar2                default null,
    p_where_clause          in varchar2                default null,
    --
    p_sql_query             in varchar2                default null,
    p_plsql_function_body   in varchar2                default null,
    --
    p_with_check_option     in boolean                 default true,
    p_optimizer_hint        in varchar2                default null,
    --
    p_dml_table_owner       in varchar2                default null,
    p_dml_table_name        in varchar2                default null,
    p_dml_plsql_code        in varchar2                default null,
    --
    p_lost_update_detection in t_lost_update_detection default null,
    p_lock_rows             in t_lock_rows             default null,
    p_lock_plsql_code       in varchar2                default null,
    --
    p_sql_parameters        in t_parameters            default c_empty_parameters ) return t_context;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_server_static_id</td>
<td>Static ID of the ORDS REST Enabled SQL Instance.</td>
</tr>
<tr>
<td>p_columns</td>
<td>DML columns to pass to the data source.</td>
</tr>
</tbody>
</table>
### Table 15-28  (Cont.) OPEN_REMOTE_DML_CONTEXT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| p_query_type               | DML columns to pass to the data source. Indicates the type of the data source: possible values are:  
  - c_query_type_table: Use a plain Table as the data source.  
  - c_query_type_sql_query: Use a SQL query as the data source.  
  - c_query_type_func_return_sql: Use the SQL query returned by the PL/SQL function. |
| p_table_owner             | For query type TABLE: Table owner                                            |
| p_table_name              | For query type TABLE: Table name                                             |
| p_where_clause            | For query type TABLE: where clause                                           |
| p_sql_query               | For query type SQL QUERY: the query                                         |
| p_plsql_function_body     | For query type PLSQL: the PL/SQL function which returns the SQL query       |
| p_with_check_option       | Specify whether to the "WITH CHECK OPTION" should be added to the data source. If set to "true" (default), INSERTED or UPDATED rows cannot violate the where clause. |
| p_optimizer_hint          | Optimizer hints to be added to the DML clause                                |
| p_dml_table_owner         | When set, DML statements will be executed against this table                |
| p_dml_table_name          | When set, DML statements will be executed against this table                |
| p_dml_plsql_code          | Custom PL/SQL code to be executed instead of DML statements                 |
| p_lost_update_detection   | lost-update detection type. Possible values are:  
  - c_lost_update_implicit: APEX calculates a checksum from the row values  
  - c_lost_update_explicit: One of the p_columns has the "is_checksum" attribute set  
  - c_lost_update_none: No lost update detection                              |
| p_lock_rows               | Specify whether to lock the rows for the (short) time frame between the lost update detection and the actual DML statement. Possible values are:  
  - c_lock_rows_automatic: use a SELECT .. FOR UPDATE  
  - c_lock_rows_plsql: use custom PL/SQL code to lock the rows  
  - c_lock_rows_none: do not lock rows                                         |
| p_dml_plsql_code          | Custom PL/SQL code to be used to lock the rows                              |
| p_sql_parameters          | Bind variables to be used                                                   |
Example

The following inserts one row into the EMP table on a REST Enabled SQL Service.

```
declare
    l_columns        apex_exec.t_columns;
    l_context        apex_exec.t_context;
begin
    -- I. Define DML columns
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'EMPNO',
        p_data_type      => apex_exec.c_data_type_number,
        p_is_primary_key => true );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'ENAME',
        p_data_type      => apex_exec.c_data_type_varchar2 );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'JOB',
        p_data_type      => apex_exec.c_data_type_varchar2 );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'HIREDATE',
        p_data_type      => apex_exec.c_data_type_date );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'MGR',
        p_data_type      => apex_exec.c_data_type_number );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'SAL',
        p_data_type      => apex_exec.c_data_type_number );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'COMM',
        p_data_type      => apex_exec.c_data_type_number );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'DEPTNO',
        p_data_type      => apex_exec.c_data_type_number );
    -- II. Open the context object
    l_context := apex_exec.open_remote_dml_context(
        p_server_static_id      => '{remote server static id}',
        p_columns               => l_columns,
        p_query_type            => apex_exec.c_query_type_sql_query,
        p_sql_query             => 'select * from emp where deptno = 10',
        p_lost_update_detection => apex_exec.c_lost_update_none );
    -- III. Provide DML data
    apex_exec.add_dml_row(
```
OPEN_REMOTE_SQL_QUERY Function

15.31 OPEN_REMOTE_SQL_QUERY Function

This function opens a query context and executes the provided SQL query on the ORDS REST Enabled SQL instance.

Syntax

function open_remote_sql_query(
    p_server_static_id in varchar2,

    p_context => l_context,
    p_operation => apex_exec.c_dml_operation_insert );

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 1,
    p_value => 4711 );

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 2,
    p_value => 'DOE' );

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 3,
    p_value => 'DEVELOPR' );

apex_exec.set_value(
    p_context => l_context,
    p_column_position => 4,
    p_value => sysdate );

apex_exec.set_value(  
    p_column_position => 6,
    p_value => 1000 );

apex_exec.set_value(  
    p_column_position => 8,
    p_value => 10 );

-- IV: Execute the DML statement

apex_exec.execute_dml(  
    p_context => l_context,
    p_continue_on_error => false);

apex_exec.close( l_context );

exception

when others then

    apex_exec.close( l_context );
    raise;

end;

Returns

The context object representing the DML handle.
Parameters

Table 15-29 OPEN_REMOTE_SQL_QUERY Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_server_static_id</td>
<td>Static ID of the ORDS REST Enabled SQL Instance.</td>
</tr>
<tr>
<td>p_sql_query</td>
<td>SQL Query to execute.</td>
</tr>
<tr>
<td>p_sql_parameters</td>
<td>Bind variables to pass to the remote server.</td>
</tr>
<tr>
<td>p_auto_bind_items</td>
<td>Whether to auto-bind all page items.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>First row to be fetched from the result set.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum amount of rows to be fetched.</td>
</tr>
<tr>
<td>p_total_row_count</td>
<td>Whether to determine the total row count.</td>
</tr>
<tr>
<td>p_total_row_count_limit</td>
<td>Upper boundary for total row count computation.</td>
</tr>
</tbody>
</table>

Returns

The context object representing a cursor for the web source query.

Example

The following example assumes a REST enabled ORDS instance to be configured in Shared Components with the static ID "My_Remote_SQL_Instance". Based on that, the example executes the query on the remote server and prints out the result set. This example code could be used within a plug-in or within a "Execute PL/SQL" region.

declare
   l_context apex_exec.t_context;

   l_idx_empno   pls_integer;
   l_idx_ename   pls_integer;
   l_idx_job     pls_integer;
   l_idx_hiredate pls_integer;
   l_idx_mgr     pls_integer;
   l_idx_sal     pls_integer;
   l_idx_comm    pls_integer;
   l_idx_deptno  pls_integer;

begin
   l_context := apex_exec.open_remote_sql_query( 
      p_server_static_id => 'My_Remote_SQL_Instance', 
      p_sql_query       => 
         'SELECT * FROMewoodb.employees WHERE empno = :empno', 
      p_sql_parameters  => 
         '[empno => :empno]', 
      p_auto_bind_items => true, 
      p_first_row       => 1, 
      p_max_rows        => 10, 
      p_total_row_count => false, 
      p_total_row_count_limit => 100 ); 
end;
OPEN_QUERYCONTEXT Procedure

This procedure is for Plug-In developers to open a query context based on the current region source. All Data Source information will be retrieved from the Plug-In region metadata.

Syntax

function open_query_context ( p_columns               in t_columns    default c_empty_columns,  
                         p_filters               in t_filters    default c_empty_filters,  
                         p_order_bys             in t_order_bys  default c_empty_order_bys,  
                         p_first_row             in pls_integer  default null,  
                         p_max_rows              in pls_integer  default null,  
                         p_total_row_count       in boolean      default false,  
                         p_total_row_count_limit in number       default null,  
                         p_sql_parameters        in t_parameters default c_empty_parameters;  

15.32 OPEN_QUERY_CONTEXT Procedure

This procedure is for Plug-In developers to open a query context based on the current region source. All Data Source information will be retrieved from the Plug-In region metadata.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_columns</td>
<td>Columns to be selected.</td>
</tr>
<tr>
<td>p_filters</td>
<td>Filters to be passed to the query context.</td>
</tr>
<tr>
<td>p_order_bys</td>
<td>Order by expressions to be passed to the query context.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>First row to be fetched from the result set.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum amount of rows to be fetched.</td>
</tr>
<tr>
<td>p_total_row_count</td>
<td>Whether to determine the total row count.</td>
</tr>
<tr>
<td>p_total_row_count_limit</td>
<td>Upper boundary for total row count computation.</td>
</tr>
<tr>
<td>p_sql_parameters</td>
<td>Additional bind variables to be used for the SQL query.</td>
</tr>
</tbody>
</table>

15.33 OPEN_QUERY_CONTEXT Function

This function opens a query context for a local database, remote database or a web source module.

Syntax

```sql
function open_query_context(
    p_location               in apex_exec_api.t_location,
    --
    p_table_owner            in varchar2                       default null,
    p_table_name             in varchar2                       default null,
    p_where_clause           in varchar2                       default null,
    p_order_by_clause        in varchar2                       default null,
    p_include_rowid_column   in boolean                        default false,
    --
    p_sql_query              in varchar2                       default null,
    p_plsql_function_body    in varchar2                       default null,
    p_optimizer_hint         in varchar2                       default null,
    --
    p_server_static_id       in varchar2                       default null,
    --
    p_module_static_id       in varchar2                       default null,
    p_web_src_parameters     in t_parameters                   default c_empty_parameters,
    p_external_filter_expr   in varchar2                       default null,
) return apex_exec_api.t_query_context;
```

Chapter 15
OPEN_QUERY_CONTEXT Function

15-45
null,
    p_external_order_by_expr in varchar2                       default
null,
    --
    p_sql_parameters in t_parameters default
c_empty_parameters,
    p_auto_bind_items in boolean   default
true,
    --
    p_columns in t_columns    default
c_empty_columns,
    --
    p_filters in t_filters    default
c_empty_filters,
    p_order_bys in t_order_bys default
c_empty_order_bys,
    --
    p_first_row in pls_integer default
null,
    p_max_rows in pls_integer default
null,
    --
    p_total_row_count in boolean default
false,
    p_total_row_count_limit in number default
null ) return t_context;

Parameters

Table 15-31  OPEN_QUERY_CONTEXT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_location</td>
<td>Location to open the query context for. Can be local database, remote database or Web Source Module. Use the C_LOCATION_LOCAL_DB, C_LOCATION_REMOTE_DB or C_LOCATION_WEB_SOURCE constants.</td>
</tr>
<tr>
<td>p_module_static_id</td>
<td>Static ID of the Web Source Module, when C_LOCATION_WEB_SOURCE has been used for p_location.</td>
</tr>
<tr>
<td>p_server_static_id</td>
<td>Static ID of the Remote Server, when C_LOCATION_REMOTE_DB has been used for p_location.</td>
</tr>
<tr>
<td>p_table_owner</td>
<td>Table owner when query type TABLE is used.</td>
</tr>
<tr>
<td>p_table_name</td>
<td>Table name when query type TABLE is used.</td>
</tr>
<tr>
<td>p_where_clause</td>
<td>Where clause to append when query type TABLE is used.</td>
</tr>
<tr>
<td>p_order_by_clause</td>
<td>Order by clause to append when query type TABLE is used.</td>
</tr>
<tr>
<td>p_include_rowid_column</td>
<td>Add the ROWID column to the SELECT list when query type TABLE is used. Defaults to false.</td>
</tr>
<tr>
<td>p_sql_query</td>
<td>SQL Query to execute when query type SQL Query is used.</td>
</tr>
<tr>
<td>p_plsql_function_body</td>
<td>PL/SQL function body returning SQL query.</td>
</tr>
</tbody>
</table>
Table 15-31  OPEN_QUERY_CONTEXT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_optimizer_hint</td>
<td>Optimizer hint to be applied to the most outer SQL query generated by APEX.</td>
</tr>
<tr>
<td>p_external_filter_expr</td>
<td>External filter expression to be passed to a Web Source Module.</td>
</tr>
<tr>
<td>p_external_order_by_expr</td>
<td>External order by expression to be passed to a Web Source Module.</td>
</tr>
<tr>
<td>p_web_src_parameters</td>
<td>Parameters to be passed to a web source module.</td>
</tr>
<tr>
<td>p_auto_bind_items</td>
<td>Whether to auto-bind APEX items (page and application items).</td>
</tr>
<tr>
<td>p_sql_parameters</td>
<td>Additional bind variables to be used for the SQL query.</td>
</tr>
<tr>
<td>p_filters</td>
<td>Filters to be passed to the query context.</td>
</tr>
<tr>
<td>p_order_bys</td>
<td>Order by expressions to be passed to the query context.</td>
</tr>
<tr>
<td>p_columns</td>
<td>Columns to be selected.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>First row to be fetched from the result set.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum amount of rows to be fetched.</td>
</tr>
<tr>
<td>p_total_row_count</td>
<td>Whether to determine the total row count.</td>
</tr>
<tr>
<td>p_total_row_count_limit</td>
<td>Upper boundary for total row count computation.</td>
</tr>
</tbody>
</table>

Returns

The context object representing a "cursor" for the query.

Example

The following example executes a query and prints out the result set. This example code can be used within a Execute PL/SQL region.

```plsql
declare
  l_context apex_exec.t_context;
  --
  l_idx_empno    pls_integer;
  l_idx_ename    pls_integer;
  l_idx_job      pls_integer;
  l_idx_hiredate pls_integer;
  l_idx_mgr      pls_integer;
  l_idx_sal      pls_integer;
  l_idx_comm     pls_integer;
  l_idx_deptno   pls_integer;
  --
  begin
    l_context := apex_exec.open_query_context(
      p_location   => apex_exec.c_location_local_db,
      p_sql_query  => 'select * from emp');
    --
    l_idx_empno := apex_exec.get_column_position( l_context, 'EMPNO');
    l_idx_ename := apex_exec.get_column_position( l_context, 'ENAME');
  end;
```

Chapter 15
OPEN_QUERY_CONTEXT Function
l_idx_job := apex_exec.get_column_position( l_context, 'JOB');
l_idx_hiredate := apex_exec.get_column_position( l_context, 'HIREDATE');
l_idx_mgr := apex_exec.get_column_position( l_context, 'MGR');
l_idx_sal := apex_exec.get_column_position( l_context, 'SAL');
l_idx_comm := apex_exec.get_column_position( l_context, 'COMM');
l_idx_deptno := apex_exec.get_column_position( l_context, 'DEPTNO');

--
while apex_exec.next_row( l_context ) loop

    htp.p( 'EMPNO: ' || apex_exec.get_number( l_context, l_idx_empno ) );
    htp.p( 'ENAME: ' || apex_exec.get_varchar2( l_context, l_idx_ename ) );
    htp.p( 'MGR:   ' || apex_exec.get_number( l_context, l_idx_mgr ) );

    end loop;

--
apex_exec.close( l_context );
return;
exception
when others then
    apex_exec.close( l_context );
    raise;
end;

15.34 OPEN_WEB__SOURCE_DML_CONTEXT Function

This function opens a DML context based for a web source module.

Syntax

function open_web_source_dml_context ( 
    p_module_static_id      in varchar2,
    p_parameters            in t_parameters            default c_empty_parameters,
    --
    p_columns               in t_columns               default c_empty_columns,
    p_lost_update_detection in t_lost_update_detection default null ) 
return t_context;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_module_static_id</td>
<td>Static ID of the web source module to use. This web source module must have operations for at least one of the Insert Rows, Update Rows or Delete rows database actions.</td>
</tr>
<tr>
<td>p_parameters</td>
<td>web source parameter values to pass to the DML context.</td>
</tr>
</tbody>
</table>
Table 15-32 (Cont.) OPEN_WEB__SOURCE_DML_CONTEXT Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_columns</td>
<td>DML columns to pass to the data source</td>
</tr>
<tr>
<td>p_lost_update_detection</td>
<td>lost-update detection type. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• c_lost_update_implicit: APEX calculates a checksum from the row values</td>
</tr>
<tr>
<td></td>
<td>• c_lost_update_explicit: One of the p_columns has the &quot;is_checksum&quot; attribute set</td>
</tr>
<tr>
<td></td>
<td>• c_lost_update_none: No lost update detection</td>
</tr>
</tbody>
</table>

Example

The following inserts one row into the EMP web source module.

declare
  l_columns        apex_exec.t_columns;
  l_context        apex_exec.t_context;
begin
  -- I. Define DML columns
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'EMPNO',
    p_data_type      => apex_exec.c_data_type_number,
    p_is_primary_key => true );
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'ENAME',
    p_data_type      => apex_exec.c_data_type_varchar2 );
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'JOB',
    p_data_type      => apex_exec.c_data_type_varchar2 );
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'HIREDATE',
    p_data_type      => apex_exec.c_data_type_date );
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'MGR',
    p_data_type      => apex_exec.c_data_type_number );
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'SAL',
    p_data_type      => apex_exec.c_data_type_number );
  apex_exec.add_column(
    p_columns        => l_columns,
    p_column_name    => 'COMM',
    p_data_type      => apex_exec.c_data_type_number );
  apex_exec.add_column(
    p_columns        => l_columns,
p_column_name => 'DEPTNO',
p_data_type => apex_exec.c_data_type_number);

-- II: Open the context object
l_context := apex_exec.open_web_source_dml_context(
p_server_static_id => '{module static id}',
p_columns => l_columns,
p_lost_update_detection => apex_exec.c_lost_update_none);

-- III: Provide DML data
apex_exec.add_dml_row(
p_context => l_context,
p_operation => apex_exec.c_dml_operation_insert);

apex_exec.set_value(
p_context => l_context,
p_column_position => 1,
p_value => 4711);

apex_exec.set_value(
p_context => l_context,
p_column_position => 2,
p_value => 'DOE' );

apex_exec.set_value(
p_context => l_context,
p_column_position => 3,
p_value => 'DEVELOPR' );

apex_exec.set_value(
p_context => l_context,
p_column_position => 4,
p_value => sysdate);

apex_exec.set_value(
p_column_position => 6,
p_value => 1000);

apex_exec.set_value(
p_context => l_context,
p_column_position => 8,
p_value => 10);

-- IV: Execute the DML statement
apex_exec.execute_dml(
p_context => l_context,
p_continue_on_error => false);

apex_exec.close( l_context );
exception
  when others then
    apex_exec.close( l_context );
    raise;
end;
Returns
The context object representing the DML handle.

15.35 OPEN_WEB_SOURCE_QUERY Function

This function opens a Web Source query context. Based on the provided web source static ID, the operation matched to the FETCH_COLLECTION database operation will be selected.

Syntax

```sql
function open_web_source_query(
    p_module_static_id       in varchar2,
    p_parameters             in t_parameters default c_empty_parameters,
    --
    p_filters                in t_filters    default c_empty_filters,
    p_order_bys              in t_order_bys  default c_empty_order_bys,
    p_columns                in t_columns    default c_empty_columns,
    --
    p_first_row              in pls_integer  default null,
    p_max_rows               in pls_integer  default null,
    --
    p_external_filter_expr   in varchar2     default null,
    p_external_order_by_expr in varchar2     default null,
    p_total_row_count        in boolean      default false )
return t_context;
```

Parameters

Table 15-33 OPEN_WEB_SOURCE_QUERY Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_module_static_id</td>
<td>Static ID of the web source module to invoke.</td>
</tr>
<tr>
<td>p_parameters</td>
<td>Parameter values to be passed to the web source.</td>
</tr>
<tr>
<td>p_filters</td>
<td>Filters to be passed to the web source.</td>
</tr>
<tr>
<td>p_order_bys</td>
<td>Order by expressions to be passed to the web source.</td>
</tr>
<tr>
<td>p_columns</td>
<td>Columns to be selected from the web source.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>First row to be fetched from the web source.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum amount of rows to be fetched from the web source.</td>
</tr>
<tr>
<td>p_external_filter_expr</td>
<td>Filter expression to be passed 1:1 to the external web service. Depends on the actual web service being used.</td>
</tr>
<tr>
<td>p_external_order_by_expr</td>
<td>Order by expression to be passed 1:1 to the external web service. Depends on the actual web service being used.</td>
</tr>
<tr>
<td>p_total_row_count</td>
<td>whether to determine the total row count (only supported when the &quot;allow fetch all rows&quot; attribute is set to Yes).</td>
</tr>
</tbody>
</table>
Returns

The context object representing a "cursor" for the web source query

Example

The following example assumes a Web Source module with the static ID "USGS" to be created in Shared Components, based on the URL endpoint https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_day.geojson. The example invokes the REST service and prints out the result set. This example code could be used within a plug-in or within a "Execute PL/SQL" region.

```plsql
declare
    l_context apex_exec.t_context;
    l_filters apex_exec.t_filters;
    l_columns apex_exec.t_columns;
    l_row     pls_integer := 1;
    l_magidx  pls_integer;
    l_titidx  pls_integer;
    l_plcidx  pls_integer;
    l_timidx  pls_integer;
    l_ididx   pls_integer;
begin
    l_context := apex_exec.open_web_source_query(
        p_module_static_id => 'USGS',
        p_max_rows         => 1000 );

    l_titidx := apex_exec.get_column_position( l_context, 'TITLE' );
    l_magidx := apex_exec.get_column_position( l_context, 'MAG' );
    l_plcidx := apex_exec.get_column_position( l_context, 'PLACE' );
    l_timidx := apex_exec.get_column_position( l_context, 'TIME' );
    l_ididx  := apex_exec.get_column_position( l_context, 'ID' );

    while apex_exec.next_row( l_context ) loop
        htp.p( 'ID:    ' || apex_exec.get_varchar2( l_context, l_ididx ) );
        htp.p( 'MAG:   ' || apex_exec.get_varchar2( l_context, l_magidx ) );
        htp.p( 'PLACE: ' || apex_exec.get_varchar2( l_context, l_plcidx ) );
        htp.p( 'TITLE: ' || apex_exec.get_varchar2( l_context, l_titidx ) );
        htp.p( 'TIME:  ' || apex_exec.get_varchar2( l_context, l_timidx ) );
    end loop;

    apex_exec.close( l_context );
exception
    when others then
        apex_exec.close( l_context );
end;
```
15.36 PURGE_WEB_SOURCE_CACHE Procedure

This procedure purges the local cache for a Web Source module. The web source module must exist in the current application and identified by its static ID. If caching is disabled or no cache entries exist, nothing happens.

Syntax

```sql
procedure purge_web_source_cache(
    p_module_static_id     in varchar2,
    p_current_session_only in boolean default false );
```

Parameters

**Table 15-34  PURGE_WEB_SOURCE_CACHE Procedure Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_module_static_id</td>
<td>Static ID of the web source module to invoke.</td>
</tr>
<tr>
<td>p_current_session_only</td>
<td>Specify true if you only want to purge entries that were saved for the current session. Defaults to false.</td>
</tr>
</tbody>
</table>

Example

Purge cache for the Web Source Module with static ID "USGS".

```sql
begin
    apex_exec.purge_web_source_cache(
        p_module_static_id => 'USGS'
    );
end;
```

15.37 SET_NULL Procedure

This procedure sets procedures to set a DML column value to NULL. Useful when the row is initialized from a query context with `set_values` and the new value of one of the columns should be NULL.

Syntax

**Signature 1**

```sql
procedure set_null(
    p_context               in t_context,
    p_column_position       in pls_integer );
```
Signature 2

procedure set_null(
    p_context           in t_context,
    p_column_name       in varchar2);

Parameters

Table 15-35  SET_NULL Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_column_position</td>
<td>Position of the column to set the value for within the DML context.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Name of the column to set the value.</td>
</tr>
</tbody>
</table>

Examples

Example 1

apex_exec.set_null(
    p_context => l_dml_context,
    p_column_position => 6);

Example 2

apex_exec.set_null(
    p_context => l_dml_context,
    p_column_name => 'SAL');

15.38 SET_VALUE Procedure

This procedure sets DML column values for different data types. To be called after add_dml_row for each column value to be set. Each procedure is called either with the column name or with the column position.

Syntax

procedure set_value(
    p_context           in t_context,
    p_column_position   in pls_integer,
    p_value             in varchar2);

procedure set_value(
    p_context           in t_context,
    p_column_name       in varchar2,
    p_value             in varchar2);
Signature 1

procedure set_value(
  p_context               in t_context,
  p_column_position       in pls_integer,
  p_value                 in number );

procedure set_value(
  p_context               in t_context,
  p_column_name           in varchar2,
  p_value                 in number );

Signature 2

procedure set_value(
  p_context               in t_context,
  p_column_position       in pls_integer,
  p_value                 in date );

procedure set_value(
  p_context               in t_context,
  p_column_name           in varchar2,
  p_value                 in date );

Signature 3

procedure set_value(
  p_context               in t_context,
  p_column_position       in pls_integer,
  p_value                 in timestamp );

procedure set_value(
  p_context               in t_context,
  p_column_name           in varchar2,
  p_value                 in timestamp );

Signature 4

procedure set_value(
  p_context               in t_context,
  p_column_position       in pls_integer,
  p_value                 in timestamp with time zone);
Signature 5

procedure set_value(
    p_context               in t_context,
    p_column_position       in pls_integer,
    p_value                 in timestamp with local time zone);

procedure set_value(
    p_context               in t_context,
    p_column_name           in varchar2,
    p_value                 in timestamp with local time zone);

Signature 6

procedure set_value(
    p_context               in t_context,
    p_column_position       in pls_integer,
    p_value                 in dsinterval_unconstrained );

procedure set_value(
    p_context               in t_context,
    p_column_name           in varchar2,
    p_value                 in dsinterval_unconstrained );

Signature 7

procedure set_value(
    p_context               in t_context,
    p_column_position       in pls_integer,
    p_value                 in yminterval_unconstrained );

procedure set_value(
    p_context               in t_context,
    p_column_name           in varchar2,
    p_value                 in yminterval_unconstrained );

Signature 8

procedure set_value(
    p_context               in t_context,
    p_column_position       in pls_integer,
    p_value                 in clob );

procedure set_value(
    p_context               in t_context,
    p_column_name           in varchar2,
    p_value                 in clob );
Signature 9

procedure set_value(
    p_context               in t_context,
    p_column_position       in pls_integer,
    p_value                 in blob);

procedure set_value(
    p_context               in t_context,
    p_column_name           in varchar2,
    p_value                 in blob);

Signature 10

procedure set_value(
    p_context               in t_context,
    p_column_position       in pls_integer,
    p_value                 in sys.anydata);

procedure set_value(
    p_context               in t_context,
    p_column_name           in varchar2,
    p_value                 in sys.anydata);

Parameters

Table 15-36 SET_VALUE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_column_position</td>
<td>Position of the column to set the value for within the DML context.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Name of the column to set the value for.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value to set.</td>
</tr>
</tbody>
</table>

Example

apex_exec.set_value(
    p_context => l_dml_context,
    p_column_name => 'SAL',
    p_value => 9500 );

apex_exec.set_value(
    p_context => l_dml_context,
    p_column_position => 6,
    p_value => 9500 );

apex_exec.set_value(
    p_context => l_dml_context,
15.39 SET_VALUES Procedure

This procedure sets all column values in the DML context with corresponding column values from the source (query) context. Useful for querying a row, changing only single columns and writing the row back.

Syntax

```sql
procedure set_values(
    p_context               in t_context,
    p_source_context        in t_context );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_source_context</td>
<td>Query context object to get column values from.</td>
</tr>
</tbody>
</table>

Example

See "SET_ROW_VERSION_CHECKSUM Procedure "

15.40 SET_ROW_VERSION_CHECKSUM Procedure

This procedure sets the row version checksum to use for lost update detection for the current DML row. This is called after add_dml_row.

Syntax

```sql
procedure set_row_version_checksum(
    p_context               in t_context,
    p_checksum              in varchar2 );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_context</td>
<td>Context object obtained with one of the OPEN_ functions.</td>
</tr>
<tr>
<td>p_checksum</td>
<td>checksum to use for lost-update detection of this row.</td>
</tr>
</tbody>
</table>
Example

The following example opens a query context on the EMP table and retrieves all values and the row version checksum for the row with EMPNO=7839. Then a DML context is opened to update the SAL column while using the row version checksum for lost update detection.

```
declare
    l_columns        apex_exec.t_columns;
    l_dml_context    apex_exec.t_context;
    l_query_context  apex_exec.t_context;
begin
    -- I. Define DML columns
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'EMPNO',
        p_data_type      => apex_exec.c_data_type_number,
        p_is_primary_key => true );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'ENAME',
        p_data_type      => apex_exec.c_data_type_varchar2 );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'JOB',
        p_data_type      => apex_exec.c_data_type_varchar2 );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'HIREDATE',
        p_data_type      => apex_exec.c_data_type_date );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'MGR',
        p_data_type      => apex_exec.c_data_type_number );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'SAL',
        p_data_type      => apex_exec.c_data_type_number );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'COMM',
        p_data_type      => apex_exec.c_data_type_number );
    apex_exec.add_column(
        p_columns        => l_columns,
        p_column_name    => 'DEPTNO',
        p_data_type      => apex_exec.c_data_type_number );

    -- II. Open the Query Context object
    l_query_context := apex_exec.open_remote_sql_query(
        p_server_static_id  => 'DevOps_Remote_SQL',
        p_sql_query         => 'select * from emp where empno = 7839',
        p_columns           => l_columns );

    -- III. Open the DML context object
```
l_dml_context := apex_exec.open_remote_dml_context(
    p_server_static_id => '(remote server static id)',
    p_columns => l_columns,
    p_query_type => apex_exec.c_query_type_sql_query,
    p_sql_query => 'select * from emp where deptno = 10',
    p_lost_update_detection => apex_exec.c_lost_update_implicit);

if apex_exec.next_row( p_context => l_query_context ) then
    apex_exec.add_dml_row(
        p_context => l_dml_context,
        p_operation => apex_exec.c_dml_operation_update);

    apex_exec.set_row_version_checksum(
        p_context => l_dml_context,
        p_checksum => apex_exec.get_row_version_checksum( p_context => l_query_context );

    apex_exec.set_values(
        p_context => l_dml_context,
        p_cource_context => l_query_context );

    apex_exec.set_value(
        p_column_name => 'SAL',
        p_value => 8000 );
else
    raise_application_error( -20000, 'EMPNO #4711 is not present!');
end if;

apex_exec.execute_dml(
    p_context => l_dml_context,
    p_continue_on_error => false);

apex_exec.close( l_dml_context );
apex_exec.close( l_query_context );
exception
    when others then
        apex_exec.close( l_dml_context );
        apex_exec.close( l_query_context );
        raise;
end;
The APEX_EXPORT package provides APIs to export the definitions of applications, files, feedback, and workspaces to text files. APEX_EXPORT uses utility types APEX_T_EXPORT_FILE and APEX_T_EXPORT_FILES. The APEX_T_EXPORT_FILE is a tuple of (name, contents) where name is the file name and contents is a clob containing the export object's definition. APEX_T_EXPORT_FILES is a table of APEX_T_EXPORT_FILE.

- **GET_APPLICATION Function**
- **GET_WORKSPACE_FILES Function**
- **GET_FEEDBACK Function**
- **GET_WORKSPACE Function**

### 16.1 GET_APPLICATION Function

This function exports the given application. Optionally, split the application definition into multiple files. The optional `p_with_%` parameters can be used to include additional information in the export.

#### Syntax

```plaintext
function get_application (  
    p_application_id          in number,  
    p_split                   in boolean  default false,  
    p_with_date               in boolean  default false,  
    p_with_ir_public_reports  in boolean  default false,  
    p_with_ir_private_reports in boolean  default false,  
    p_with_ir_notifications   in boolean  default false,  
    p_with_translations       in boolean  default false,  
    p_with_pkg_app_mapping    in boolean  default false,  
    p_with_original_ids       in boolean  default false,  
    p_with_no_subscriptions   in boolean  default false,  
    p_with_comments           in boolean  default false,  
    p_with_supporting_objects in varchar2 default null,  
    p_with_acl_assignments    in boolean  default false)  
return wwv_flow_t_export_files;
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application id.</td>
</tr>
<tr>
<td>p_split</td>
<td>If true, split the definition into discrete elements that can be stored in separate files. If false, the result is a single file.</td>
</tr>
</tbody>
</table>
Table 16-1  (Cont.) GET_APPLICATION Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_with_date</td>
<td>If true, include export date and time in the result.</td>
</tr>
<tr>
<td>p_with_public_reports</td>
<td>If true, include public reports that a user saved.</td>
</tr>
<tr>
<td>p_with_private_reports</td>
<td>If true, include private reports that a user saved.</td>
</tr>
<tr>
<td>p_with_notifications</td>
<td>If true, include report notifications.</td>
</tr>
<tr>
<td>p_with_translations</td>
<td>If true, include application translation mappings and all text from the translation repository.</td>
</tr>
<tr>
<td>p_with_pkg_app_mapping</td>
<td>If true, export installed packaged applications with references to the packaged application definition. If false, export them as normal applications.</td>
</tr>
<tr>
<td>p_with_original_ids</td>
<td>If true, export with the IDs as they were when the application was imported.</td>
</tr>
<tr>
<td>p_with_no_subscription</td>
<td>If true, components contain subscription references.</td>
</tr>
<tr>
<td>p_with_comments</td>
<td>If true, include developer comments.</td>
</tr>
<tr>
<td>p_withSupportingObjects</td>
<td>If 'Y', export supporting objects. If 'I', automatically install on import. If 'N', do not export supporting objects. If null, the application's include in export deployment value is used.</td>
</tr>
<tr>
<td>p_with_acl_assignments</td>
<td>If true, export ACL user role assignments.</td>
</tr>
</tbody>
</table>

RETURNS

A table of apex_t_export_file. Unless the caller passes p_split=>true to the function, the result will be a single file.

Example

This sqlplus code fragment spools the definition of application 100 into file f100.sql.

```
variable name varchar2(255)
variable contents clob
declare
  l_files apex_t_export_files;
begin
  l_files := apex_export.get_application(p_application_id => 100);
  :name := l_files(1).name;
  :contents := l_files(1).contents;
end;
/
set feed off echo off head off flush off termout off trimspool on
set long 100000000 longchunksize 32767
col name new_val name
select :name name from sys.dual;
spool &name.
print contents
spool off
```
16.2 GET_WORKSPACE_FILES Function

This function exports the given workspace’s static files.

Syntax

function get_workspace_files (  
    p_workspace_id in number,  
    p_with_date   in boolean default false )  
return wwv_flow_t_export_files;

Parameters

Table 16-2  GET_WORKSPACE_FILES Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The workspace id.</td>
</tr>
<tr>
<td>p_with_date</td>
<td>If true, include export date and time in the result.</td>
</tr>
</tbody>
</table>

RETURNS

A table of apex_t_export_file. The result is a single file, splitting into multiple files will be implemented in a future release.

Example

Export the workspace files of the workspace with id 12345678.

declare  
    l_file apex_t_export_files;
begin  
    l_file := apex_export.get_workspace(p_workspace_id => 12345678);
end;

16.3 GET_FEEDBACK Function

This function exports user feedback to the development environment or developer feedback to the deployment environment.

Syntax

function get_feedback (  
    p_workspace_id      in number,  
    p_with_date         in boolean  default false,  
    p_since             in date     default null,  
    p_deployment_system in varchar2 default null )  
return wwv_flow_t_export_files;
Parameters

Table 16-3 GET_FEEDBACK Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The workspace id.</td>
</tr>
<tr>
<td>p_with_date</td>
<td>If true, include export date and time in the result.</td>
</tr>
<tr>
<td>p_since</td>
<td>If set, only export feedback that has been gathered since the given date.</td>
</tr>
<tr>
<td>p_deployment_system</td>
<td>If null, export user feedback. If not null, export developer feedback for the given deployment system.</td>
</tr>
</tbody>
</table>

RETURNS

A table of apex_t_export_file.

Examples

Example 1

Export feedback to development environment.

```declare
    l_file apex_t_export_files;
begin
    l_file := apex_export.get_feedback(p_workspace_id => 12345678);
end;```

Example 2

Export developer feedback in workspace 12345678 since 8-MAR-2010 to deployment environment EA2.

```declare
    l_file apex_t_export_files;
begin
    l_file := apex_export.get_feedback(
        p_workspace_id => 12345678,
        p_since => date'2010-03-08',
        p_deployment_system => 'EA2' );
end;```

16.4 GET_WORKSPACE Function

This function exports the given workspace's definition and users. The optional p_with_ % parameters which all default to false can be used to include additional information in the export.
Syntax

function get_workspace (
    p_workspace_id          in number,
    p_with_date             in boolean default false,
    p_with_team_development in boolean default false,
    p_with_misc             in boolean default false )
return wwv_flow_t_export_files;

Parameters

Table 16-4  GET_WORKSPACE Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The workspace id.</td>
</tr>
<tr>
<td>p_with_date</td>
<td>If true, include export date and time in the result.</td>
</tr>
<tr>
<td>p_with_team_development</td>
<td>If true, include team development data.</td>
</tr>
<tr>
<td>p_with_misc</td>
<td>If true, include data from SQL Workshop, mail logs, etc. in the export.</td>
</tr>
</tbody>
</table>

Returns

A table of apex_t_export_file.

Examples

Export the definition of workspace #12345678.

declare
    l_file apex_t_export_files;
begin
    l_files := apex_export.get_workspace(p_workspace_id => 12345678);
end;
The APEX_INSTANCE_ADMIN package provides utilities for managing an Oracle Application Express runtime environment. You use the APEX_INSTANCE_ADMIN package to get and set email settings, Oracle Wallet settings, report printing settings and to manage schema to workspace mappings. APEX_INSTANCE_ADMIN can be executed by the SYS, SYSTEM, and APEX_190100 database users and any database user granted the role APEX_ADMINISTRATOR_ROLE.

- Available Parameter Values
- ADD_SCHEMA Procedure
- ADD_WORKSPACE Procedure
- CREATE_SCHEMA_EXCEPTION Procedure
- DB_SIGNATURE Function
- FREE_WORKSPACE_APP_IDS Procedure
- GET_PARAMETER Function
- GET_SCHEMAS Function
- GET_WORKSPACE_PARAMETER
- IS_DB_SIGNATURE_VALID Function
- REMOVE_APPLICATION Procedure
- REMOVE_SAVED_REPORT Procedure
- REMOVE_SAVED_REPORTS Procedure
- REMOVE_SCHEMA Procedure
- REMOVE_SCHEMA_EXCEPTION Procedure
- REMOVE_SCHEMA_EXCEPTIONS Procedure
- REMOVE_SUBSCRIPTION Procedure
- REMOVE_WORKSPACE Procedure
- REMOVE_WORKSPACE_EXCEPTIONS Procedure
- RESERVE_WORKSPACE_APP_IDS Procedure
- RESTRICT_SCHEMA Procedure
- SET_LOG_SWITCH_INTERVAL Procedure
- SET_WORKSPACE_PARAMETER
- SET_PARAMETER Procedure
- SET_WORKSPACE_CONSUMER_GROUP Procedure
- TRUNCATE_LOG Procedure
- UNRESTRICT_SCHEMA Procedure
17.1 Available Parameter Values

Table 17-1 lists all the available parameter values you can set within the APEX_INSTANCE_ADMIN package, including parameters for email, wallet, and reporting printing.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT_LIFETIME_DAYS</td>
<td>The maximum number of days an end-user account password may be used before the account is expired.</td>
</tr>
<tr>
<td>ALLOW_DB_MONITOR</td>
<td>If set to Y, the default, database monitoring is enabled. If set to N, it is disabled.</td>
</tr>
<tr>
<td>ALLOW_HOSTNAMES</td>
<td>If set, users can only navigate to an application if the URL's hostname part contains this value. Instance administrators can configure more specific values at workspace level.</td>
</tr>
<tr>
<td>ALLOW_PUBLIC_FILE_UPLOAD</td>
<td>If set to Y, file uploads are allowed without user authentication. If set to N, the default, they are not allowed.</td>
</tr>
<tr>
<td>ALLOW_RAS</td>
<td>This parameter is only supported if running Oracle Database 12c. If set to Y, enable Real Application Security support for applications. If set to N (the default), Real Application Security cannot be used.</td>
</tr>
<tr>
<td>ALLOW_REST</td>
<td>If set to Y, the default, developers are allowed to expose report regions as RESTful services. If set to N, the are not allowed.</td>
</tr>
<tr>
<td>APEX_BUILDER_AUTHENTICATION</td>
<td>Controls the authentication scheme for the internal builder applications. Valid parameter values include:</td>
</tr>
<tr>
<td></td>
<td>• APEX - Application Express workspace accounts authentication (default)</td>
</tr>
<tr>
<td></td>
<td>• DB - Database accounts authentication</td>
</tr>
<tr>
<td></td>
<td>• HEADER - HTTP header variable based authentication</td>
</tr>
<tr>
<td></td>
<td>• SSO - Oracle Single Sign-On authentication</td>
</tr>
<tr>
<td></td>
<td>• LDAP - LDAP authentication</td>
</tr>
<tr>
<td>APEX_REST_PATH_PREFIX</td>
<td>Controls the URI path prefix used to access built-in RESTful Services exposed by Application Express. For example, built-in RESTful Service for referencing static application files using #APP_IMAGES# token. If the default prefix (r) conflicts with RESTful Services defined by users, adjust this preference to avoid the conflict.</td>
</tr>
<tr>
<td>APPLICATION_ACTIVITY_LOGGING</td>
<td>Controls instance wide setting of application activity log ([A]lways, [N]ever, [U]se application settings)</td>
</tr>
<tr>
<td>APPLICATION_ID_MAX</td>
<td>The largest possible ID for a websheet or database application.</td>
</tr>
<tr>
<td>APPLICATION_ID_MIN</td>
<td>The smallest possible ID for a websheet or database application.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AUTOEXTEND_TABLESPACES</td>
<td>If set to Y, the default, provisioned tablespaces is autoextended up to a maximum size. If set to N, tablespaces are not autoextended.</td>
</tr>
<tr>
<td>BIGFILE_TABLESPACES_ENABLED</td>
<td>If set to Y, the tablespaces provisioned through Oracle Application Express are created as bigfile tablespaces. If set to N, the tablespaces are created as smallfile tablespaces.</td>
</tr>
<tr>
<td>CHECKSUM_HASH_FUNCTION</td>
<td>Defines the algorithm that is used to create one way hashes for URL checksums. Valid values are MD5 (deprecated), SH1 (SHA-1), SH256 (SHA-2, 256 bit), SH384 (SHA-2, 384 bit), SH512 (SHA-2, 512 bit) and null. The SHA-2 algorithms are only available on Oracle Database Release 12g and later. A null value evaluates to the most secure algorithm available and is the default.</td>
</tr>
<tr>
<td>CHECK_FOR_UPDATES</td>
<td>If set to N, the check for Oracle Application Express and Oracle REST Data Services product updates is disabled for the entire instance, regardless of preferences specified by individual developers. The default is Y.</td>
</tr>
<tr>
<td>DB_SIGNATURE</td>
<td>Set to the database host/service name on install. If it differs, for example, on cloned databases, sending emails will fail. A value of null (the default) disables any checks.</td>
</tr>
<tr>
<td>DEBUG_MESSAGE_PAGE_VIEW_LIMIT</td>
<td>Maximum number of debug messages for a single page view. Default is 50000.</td>
</tr>
<tr>
<td>DELETE_UPLOADED_FILES_AFTER_DAYS</td>
<td>Uploaded files like application export files, websheet export files, spreadsheet data load files are automatically deleted after this number of days. Default is 14.</td>
</tr>
<tr>
<td>DISABLE_ADMIN_LOGIN</td>
<td>If set to Y, administration services are disabled. If set to N, the default, they are not disabled.</td>
</tr>
<tr>
<td>DISABLE_WORKSPACE_LOGIN</td>
<td>If set to Y, the workspace login is disabled. If set to N, the default, the login is not disabled.</td>
</tr>
<tr>
<td>DISABLE_WS_PROV</td>
<td>If set to Y, the workspace creation is disabled for requests sent out by using e-mail notification. If set to N, the default, they are not disabled.</td>
</tr>
</tbody>
</table>
| EMAIL_IMAGES_URL                   | Specifies the full URL to the images directory of Application Express instance, including the trailing slash after the images directory. For example: http://your_server/i/  
This setting is used for Oracle Application Express system-generated emails. |
| EMAIL_INSTANCE_URL                 | Specifies the URL to Oracle Application Express instance, including the trailing slash after the Database Access Descriptor. For example: http://your_server/pls/apex/ 
This setting used for Oracle Application Express system-generated emails. |
| ENABLE_TRANSACTIONAL_SQL           | If set to Y, the default, transactional SQL commands are enabled on this instance. If set to N, they are not enabled.                                                                                           |
### Table 17-1 (Cont.) Available Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ENCRYPTED_TABLESPACES_ENABLED</strong></td>
<td>If set to Y, the tablespaces provisioned through Oracle Application Express are created as encrypted tablespaces. If set to N, the tablespaces are not encrypted.</td>
</tr>
<tr>
<td><strong>EXPIRE_FND_USER_ACCOUNTS</strong></td>
<td>If set to Y, expiration of Application Express accounts is enabled. If set to N, they are not enabled.</td>
</tr>
<tr>
<td><strong>HEADER_AUTH_CALLBACK</strong></td>
<td>Callback procedure name for HTTP header based authentication, defaults to apex_authentication.callback.</td>
</tr>
<tr>
<td><strong>HTTP_ERROR_STATUS_ON_ERROR_PAGE_ENABLED</strong></td>
<td>Used in conjunction with the APEX_INSTANCE_ADMIN.SET_PARAMETER procedure. If set to N, the default, Oracle Application Express presents an error page to the end user for all unhandled errors. If set to Y, returns an HTTP 400 status to the end user's client browser when the Application Express engine encounters an unhandled error.</td>
</tr>
<tr>
<td><strong>HTTP_RESPONSE_HEADERS</strong></td>
<td>List of http response headers, separated by newline (chr(10)). Application Express writes these headers on each request, before rendering the page. The substitution string #CDN# within the headers is replaced with the content delivery networks that are known to Application Express.</td>
</tr>
<tr>
<td><strong>HTTP_STS_MAX_AGE</strong></td>
<td>REQUIRE_HTTPS must be set to A for this parameter to be relevant. Application Express emits a Strict-Transport-Security header, with max-age=&lt;value&gt;, on HTTPS requests if HTTP_STS_MAX_AGE has a value greater than 0. If the request protocol is HTTP, instead of processing the request, Application Express redirects to a HTTPS URL.</td>
</tr>
<tr>
<td><strong>INBOUND_PROXIES</strong></td>
<td>Comma-separated list of IP addresses for proxy servers through which requests come in.</td>
</tr>
<tr>
<td><strong>INSTANCE_PROXY</strong></td>
<td>The proxy server for all outbound HTTP(s) traffic. If INSTANCE_PROXY is set, it overrides any application specific proxy server definition.</td>
</tr>
<tr>
<td><strong>INSTANCE_NO_PROXY_DOMAINS</strong></td>
<td>Comma-separated list of domain names for which the instance proxy is not to be used.</td>
</tr>
<tr>
<td><strong>INSTANCE_TABLESPACE</strong></td>
<td>If specified, the tablespace to use for the database user for all new workspaces.</td>
</tr>
<tr>
<td><strong>KEEP_SESSIONS_ON_UPGRADE</strong></td>
<td>This flag affects application upgrades. If set to N, the default, delete sessions associated with the application. If set to Y, leave sessions unaffected.</td>
</tr>
<tr>
<td><strong>LOGIN_THROTTLE_DELAY</strong></td>
<td>The flag which determines the time increase in seconds after failed logins.</td>
</tr>
<tr>
<td><strong>LOGIN_THROTTLE_METHODS</strong></td>
<td>The methods to count failed logins. Colon-separated list of USERNAME_IP, USERNAME, IP.</td>
</tr>
<tr>
<td><strong>MAX_SESSION_IDLE_SEC</strong></td>
<td>The number of seconds an internal application may be idle.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_SESSION_IDLE_SEC</td>
<td>The number of seconds an internal application may be idle.</td>
</tr>
<tr>
<td>MAX_SESSION_LENGTH_SEC</td>
<td>The number of seconds an internal application session may exist.</td>
</tr>
<tr>
<td>MAX_WEBSERVICE_REQUESTS</td>
<td>The maximum number of outbound web service requests allowed for each workspace in a rolling 24-hour period. Default is 1000.</td>
</tr>
<tr>
<td>PASSWORD_ALPHA_CHARACTERS</td>
<td>The alphabetic characters used for password complexity rules. Default list of alphabetic characters include the following: a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z</td>
</tr>
<tr>
<td>PASSWORD_HASH_FUNCTION</td>
<td>Defines the algorithm that is used to create one way hashes for workspace user passwords. Valid values are MD5 (deprecated), SH1 (SHA-1), SH256 (SHA-2, 256 bit), SH384 (SHA-2, 384 bit), SH512 (SHA-2, 512 bit) and null. The SHA-2 algorithms are only available on Oracle Database Release 12g and later. A null value evaluates to the most secure algorithm available and is the default.</td>
</tr>
<tr>
<td>PASSWORD_HASH_ITERATIONS</td>
<td>Defines the number of iterations for the PASSWORD_HASH_FUNCTION (default 10000).</td>
</tr>
<tr>
<td>PASSWORD_HISTORY_DAYS</td>
<td>Defines the maximum number of days a developer or administrator account password may be used before the account expires. The default value is 45 days.</td>
</tr>
<tr>
<td>PASSWORD_PUNCTUATION_CHARACTERS</td>
<td>The punctuation characters used for password complexity rules. Default list of punctuation characters include the following: !&quot;#$%&amp;'()*+,-./:;&lt;=&gt;?_</td>
</tr>
<tr>
<td>PLSQL_EDITING</td>
<td>If set to Y, the default, the SQL Workshop Object Browser is enabled to allow users to edit and compile PL/SQL. If set to N, users are not allowed.</td>
</tr>
<tr>
<td>PRINT_BIB_LICENSED</td>
<td>Specify either standard support or advanced support. Advanced support requires an Oracle BI Publisher license. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>• APEX_LISTENER - Requires Oracle Rest Data Services</td>
</tr>
<tr>
<td></td>
<td>• ADVANCED - Requires Oracle BI Publisher</td>
</tr>
<tr>
<td></td>
<td>• STANDARD</td>
</tr>
<tr>
<td>PRINT_SVR_HOST</td>
<td>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.</td>
</tr>
<tr>
<td>PRINT_SVR_PORT</td>
<td>Defines the port of the print server engine, for example 8888. Value must be a positive integer.</td>
</tr>
<tr>
<td>PRINT_SVR_PROTOCOL</td>
<td>Valid values include:</td>
</tr>
<tr>
<td></td>
<td>• http</td>
</tr>
<tr>
<td></td>
<td>• https</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRINT_SVR_SCRIPT</td>
<td>Defines the script that is the print server engine, for example: /xmlpserver/convert</td>
</tr>
<tr>
<td>QOS_MAX_SESSION_KILL_TIMEOUT</td>
<td>Number of seconds that an active old session can live, when QOS_MAX_SESSION_REQUESTS has been reached. The oldest database session with LAST_CALL_ET greater than QOS_MAX_SESSION_KILL_TIMEOUT is killed.</td>
</tr>
<tr>
<td>QOS_MAX_SESSION_REQUESTS</td>
<td>Number of allowed concurrent requests to one session associated with this workspace.</td>
</tr>
<tr>
<td>QOS_MAX_WORKSPACE_REQUESTS</td>
<td>Number of allowed concurrent requests to sessions in this workspace.</td>
</tr>
<tr>
<td>REQ_NEW_SCHEMA</td>
<td>If set to Y, the option for new schema for new workspace requests is enabled. If set to N, the default, the option is disabled.</td>
</tr>
<tr>
<td>REQUIRE_HTTPS</td>
<td>Set to A, to enforce HTTPS for the entire Application Express instance. Set to I, to enforce HTTPS within the Application Express development and administration applications. Set to N, to allow all applications to be used when the protocol is either HTTP or HTTPS. Please note developers can also enforce HTTPS at the application level, by setting the Secure attribute of an application scheme's cookie.</td>
</tr>
<tr>
<td>REQUIRE_VERIFICATION_CODE</td>
<td>If set to Y, the Verification Code is displayed and is required for someone to request a new workspace. If set to N, the default, the Verification Code is not required.</td>
</tr>
<tr>
<td>RESTFUL_SERVICES_ENABLED</td>
<td>If set to Y, the default, RESTful services development is enabled. If set to N, RESTful services are not enabled.</td>
</tr>
<tr>
<td>RESTRICT_IP_RANGE</td>
<td>To restrict access to the Application Express development environment to a specific range of IP addresses, enter a comma-delimited list of IP addresses. If necessary, you can use an asterisk (<em>) as a wildcard, but do not include additional numeric values after wildcard characters. For example, 138.</em>.41.2 is not a valid value.</td>
</tr>
<tr>
<td>RM_CONSUMER_GROUP</td>
<td>If set, this is the resource manager consumer group to be used for all page events. A more specific group can be configured at workspace level.</td>
</tr>
<tr>
<td>SERVICE_REQUEST_FLOW</td>
<td>Determines default provisioning mode. Default is MANUAL.</td>
</tr>
<tr>
<td>SERVICE_REQUESTS_ENABLED</td>
<td>If set to Y, the default, workspace service requests for schemas, storage, and termination is enabled. If set to N, these requests are disabled.</td>
</tr>
<tr>
<td>SMTP_FROM</td>
<td>Defines the &quot;from&quot; address for administrative tasks that generate email, such as approving a provision request or resetting a password. Enter a valid email address, for example: <a href="mailto:someone@somewhere.com">someone@somewhere.com</a></td>
</tr>
</tbody>
</table>
### Table 17-1  (Cont.) Available Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP_HOST_ADDRESS</td>
<td>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</td>
</tr>
<tr>
<td>SMTP_HOST_PORT</td>
<td>Defines the port the SMTP server listens to for mail requests. Default setting: 25</td>
</tr>
<tr>
<td>SMTP_PASSWORD</td>
<td>Defines the password Application Express takes to authenticate itself against the SMTP server, with the parameter SMTP_USERNAME.</td>
</tr>
<tr>
<td>SMTP_TLS_MODE</td>
<td>Defines whether Application Express opens an encrypted connection to the SMTP server. Encryption is only supported on database versions 11.2.0.2 and later. On earlier database versions, the connection is not encrypted. If set to N, the connection is unencrypted (default). If set to Y, the connection is encrypted before data is sent. If STARTTLS, Application Express sends the SMTP commands EHLO &lt;SMTP_HOST_ADDRESS&gt; and STARTTLS before encrypting the connection.</td>
</tr>
<tr>
<td>SMTP_USERNAME</td>
<td>Defines the username Application Express takes to authenticate itself against the SMTP server (default is null). Starting with database version 11.2.0.2, Application Express uses UTL_MAIL'S AUTH procedure for authentication. This procedure negotiates an authentication mode with the SMTP server. With earlier database versions, the authentication mode is always AUTH LOGIN. If SMTP_USERNAME is null, no authentication is used.</td>
</tr>
<tr>
<td>SOCIAL_AUTH_CALLBACK</td>
<td>Callback procedure name for Social Sign-In, defaults to apex_authentication.callback.</td>
</tr>
<tr>
<td>SQL_SCRIPT_MAX_OUTPUT_SIZE</td>
<td>The maximum allowable size for an individual script result. Default is 200000.</td>
</tr>
<tr>
<td>SSO_LOGOUT_URL</td>
<td>Defines the URL Application Express redirects to in order to trigger a logout from the Single Sign-On server. Application Express automatically appends &quot;?p_done_url=...login url...&quot;. Example: <a href="https://login.mycompany.com/pls/orasso/orasso.wwsso_app_admin.ls_logout">https://login.mycompany.com/pls/orasso/orasso.wwsso_app_admin.ls_logout</a></td>
</tr>
<tr>
<td>STRONG_SITE_ADMIN_PASSWORD</td>
<td>If set to Y, the default, the apex_admin password must conform to the default set of strong complexity rules. If set to N, the password is not required to follow the strong complexity rules.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SYSTEM_HELP_URL</td>
<td>Location of the help and documentation accessed from the Help link within the development environment. Default is <a href="http://apex.oracle.com/doc41">http://apex.oracle.com/doc41</a>.</td>
</tr>
<tr>
<td>TRACING_ENABLED</td>
<td>If set to Y (the default), an application with Debug enabled can also generate server side db trace files using &amp;p_trace=YES on the URL. If set to N, the request to create a trace file is ignored.</td>
</tr>
<tr>
<td>USERNAME_VALIDATION</td>
<td>The regular expression used to validate a username if the Builder authentication scheme is not APEX. Default is as follows: ^([[:alnum:]_-%]+@[[:alnum:].-]+[[[:alpha:]]]{2,4}$</td>
</tr>
<tr>
<td>WALLET_PATH</td>
<td>The path to the wallet on the file system, for example: file:/home/&lt;username&gt;/wallets</td>
</tr>
<tr>
<td>WALLET_PWD</td>
<td>The password associated with the wallet.</td>
</tr>
<tr>
<td>WEBSHEET_SQL_ACCESS</td>
<td>If set to Y, the default, SQL tags and SQL reports are possible in Websheet applications. If set to N, they are not possible.</td>
</tr>
<tr>
<td>WORKSPACE_EMAIL_MAXIMUM</td>
<td>Maximum number of emails allowed to be sent by using APEX_MAIL per workspace in a 24 hour period. Default is 1000.</td>
</tr>
<tr>
<td>WORKSPACE_MAX_FILE_BYTES</td>
<td>The maximum number of bytes for uploaded files for a workspace. A setting at the workspace-level overrides the instance-level setting.</td>
</tr>
<tr>
<td>WORKSPACE_MAX_OUTPUT_SIZE</td>
<td>The maximum space allocated for script results. Default is 2000000.</td>
</tr>
<tr>
<td>WORKSPACE_PROVISION_DEMO_OBJECTS</td>
<td>If set to Y, the default, demonstration applications and database objects are created in new workspaces. If set to N, they are not created in the current workspace.</td>
</tr>
<tr>
<td>WORKSPACE_WEBSHEET_OBJECTS</td>
<td>If set to Y, the default, Application Express Websheet database objects are created in new workspaces. If set to N, they are not created in the current workspace.</td>
</tr>
<tr>
<td>WORKSPACE_TEAM_DEV_FILES_YN</td>
<td>If set to Y, the default, new workspaces will allow file uploads into Team Development. If set to N, new workspaces will not allow file uploads into Team Development, thereby disabling the ability to upload &quot;feature&quot;, &quot;bug&quot;, and &quot;feedback&quot; attachments.</td>
</tr>
<tr>
<td>WORKSPACE_TEAM_DEV_FS_LIMIT</td>
<td>The maximum per upload file size of a Team Development file (&quot;feature&quot;, &quot;bug&quot;, and &quot;feedback&quot; attachments). Default value is 15728640 (15 MB). All possible options are listed below: 5 MB - 5242880</td>
</tr>
</tbody>
</table>
17.2 ADD_SCHEMA Procedure

The ADD_SCHEMA procedure adds a schema to a workspace to schema mapping.

Syntax

APEX_INSTANCE_ADMIN.ADD_SCHEMA(  
    p_workspace IN VARCHAR2,  
    p_schema IN VARCHAR2);  

Parameters

Table 17-2  ADD_SCHEMA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The name of the workspace to which the schema mapping is added.</td>
</tr>
<tr>
<td>p_schema</td>
<td>The schema to add to the schema to workspace mapping.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the ADD_SCHEMA procedure to map a schema mapped to a workspace.

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

17.3 ADD_WORKSPACE Procedure

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

Syntax

APEX_INSTANCE_ADMIN.ADD_WORKSPACE(  
    p_workspace_id IN NUMBER DEFAULT NULL,  
    p_workspace IN VARCHAR2,  
    p_source_identifier IN VARCHAR2 DEFAULT NULL,  
    p_primary_schema IN VARCHAR2,  
);
Parameters

Table 17-3  ADD_WORKSPACE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The ID to uniquely identify the workspace in an Application Express instance. This may be left null and a new unique ID is assigned.</td>
</tr>
<tr>
<td>p_workspace</td>
<td>The name of the workspace to be added.</td>
</tr>
<tr>
<td>p_source_identifier</td>
<td>A short identifier for the workspace used when synchronizing feedback between different instances.</td>
</tr>
<tr>
<td>p_primary_schema</td>
<td>The primary database schema to associate with the new workspace.</td>
</tr>
<tr>
<td>p_additional_schemas</td>
<td>A colon delimited list of additional schemas to associate with this workspace.</td>
</tr>
<tr>
<td>p_rm_consumer_group</td>
<td>Resource Manager consumer group which is used when executing applications of this workspace.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the ADD_WORKSPACE procedure to add a new workspace named MY_WORKSPACE using the primary schema, SCOTT, along with additional schema mappings for HR and OE.

```sql
BEGIN
    APEX_INSTANCE_ADMIN.ADD_WORKSPACE (
        p_workspace_id       => 8675309,
        p_workspace          => 'MY_WORKSPACE',
        p_primary_schema     => 'SCOTT',
        p_additional_schemas => 'HR:OE');
END;
```

17.4 CREATE_SCHEMA_EXCEPTION Procedure

This procedure creates an exception which allows assignment of a restricted schema to a specific workspace.

Syntax

```sql
APEX_INSTANCE_ADMIN.CREATE_SCHEMA_EXCEPTION (  
    p_schema       in varchar2,
    p_workspace    in varchar2 );
```
### 17.5 DB_SIGNATURE Function

The **DB_SIGNATURE** function computes the current database signature value.

#### Syntax

```sql
function db_signature
    return varchar2;
```

#### Example

The following example prints the database signature:

```sql
begin
    apex_instance_admin.set_parameter (p_parameter => 'DB_SIGNATURE',
```
17.6 FREE_WORKSPACE_APP_IDS Procedure

This procedure removes the reservation of application IDs for a given workspace ID. Use this procedure to undo a reservation, when the reservation is not necessary anymore because it happened by mistake or the workspace no longer exists. To reserve application IDs for a given workspace, see "RESERVE_WORKSPACE_APP_IDS Procedure."

Syntax

APEX_INSTANCE_ADMIN.FREE_WORKSPACE_APP_IDS ( 
   p_workspace_id IN NUMBER );

Parameters

Table 17-5  FREE_WORKSPACE_APP_IDS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The unique ID of the workspace.</td>
</tr>
</tbody>
</table>

Example

This example illustrates how to undo the reservation of application IDS that belong to a workspace with an ID of 1234567890.

begin  
    apex_instance_admin.free_workspace_app_ids(1234567890);  
end;

17.7 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>
<thead>
<tr>
<th>Table 17-6</th>
<th>GET_PARAMETER Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>p_parameter</td>
<td>The instance parameter to be retrieved. See &quot;Available Parameter Values&quot;.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the GET_PARAMETER function to retrieve the SMTP_HOST_ADDRESS parameter currently defined for an Oracle Application Express instance.

```
DECLARE
  L_VAL VARCHAR2(4000);
BEGIN
  L_VAL := APEX_INSTANCE_ADMIN.GET_PARAMETER('SMTP_HOST_ADDRESS');
  DBMS_OUTPUT.PUT_LINE('The SMTP Host Setting Is: ' || L_VAL);
END;
```

17.8 GET_SCHEMAS Function

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Table 17-7</th>
<th>GET_SCHEMAS Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>p_workspace</td>
<td>The name of the workspace from which to retrieve the schema list.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the GET_SCHEMA function to retrieve the underlying schemas mapped to a workspace.

```
DECLARE
  L_VAL VARCHAR2(4000);
BEGIN
  L_VAL := APEX_INSTANCE_ADMIN.GET_SCHEMAS('MY_WORKSPACE');
```
17.9 GET_WORKSPACE_PARAMETER

The `GET_WORKSPACE_PARAMETER` procedure gets the workspace parameter.

Syntax

```sql
get_workspace_parameter(
    p_workspace     IN VARCHAR2,
    p_parameter     IN VARCHAR2,
);
```

Parameters

**Table 17-8  GET_WORKSPACE_PARAMETER Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The name of the workspace to which you are getting the workspace parameter.</td>
</tr>
<tr>
<td>p_parameter</td>
<td>The parameter name that overrides the instance parameter value of the same name for this workspace. Parameter names include:</td>
</tr>
<tr>
<td></td>
<td>• ALLOW_HOSTNAMES</td>
</tr>
<tr>
<td></td>
<td>• MAX_SESSION_IDLE_SEC</td>
</tr>
<tr>
<td></td>
<td>• MAX_SESSION_LENGTH_SEC</td>
</tr>
<tr>
<td></td>
<td>• QOS_MAX_WORKSPACE_REQUESTS</td>
</tr>
<tr>
<td></td>
<td>• QOS_MAX_SESSION_REQUESTS</td>
</tr>
<tr>
<td></td>
<td>• QOS_MAX_SESSION_KILL_TIMEOUT</td>
</tr>
<tr>
<td></td>
<td>• RM_CONSUMER_GROUP</td>
</tr>
<tr>
<td></td>
<td>• WORKSPACE_EMAIL_MAXIMUM</td>
</tr>
<tr>
<td></td>
<td>• WORKSPACE_MAX_FILE_BYTES</td>
</tr>
</tbody>
</table>

Example

The following example prints the value of `ALLOW_HOSTNAMES` for the HR workspace.

```
BEGIN
    DBMS_OUTPUT.PUT_LINE ( APEX_INSTANCE_ADMIN.GET_WORKSPACE_PARAMETER ( p_workspace => 'HR', p_parameter => 'ALLOW_HOSTNAMES' ));
END;
```

17.10 IS_DB_SIGNATURE_VALID Function

The `IS_DB_SIGNATURE_VALID` function returns whether the instance parameter `DB_SIGNATURE` matches the value of the function `db_signature`. If the instance parameter is not set (the default), also return `true`. 
Syntax

function is_db_signature_valid
  return boolean;

Example

The following example prints the signature is valid.

begin
  sys.dbms_output.put_line ( 
    case when apex_instance_admin.is_db_signature_valid 
    then 'signature is valid, features are enabled' 
    else 'signature differs (cloned db), features are disabled' 
    end );
end;

See Also:

"DB_SIGNATURE Function", "Available Parameter Values"

17.11 REMOVE_APPLICATION Procedure

The REMOVE_APPLICATION procedure removes the application specified from the Application Express instance.

Syntax

APEX_INSTANCE_ADMIN.REMOVE_APPLICATION ( 
    p_application_id IN NUMBER);

Parameters

Table 17-9  REMOVE_APPLICATION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application to remove.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the REMOVE_APPLICATION procedure to remove an application with an ID of 100 from an Application Express instance.

BEGIN
  APEX_INSTANCE_ADMIN.REMOVE_APPLICATION(100);
END;
17.12 REMOVE_SAVED_REPORT Procedure

The REMOVE_SAVED_REPORT procedure removes a specific user's saved interactive report settings for a particular application.

Syntax

APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(
    p_application_id    IN NUMBER,
    p_report_id         IN NUMBER);

Parameters

Table 17-10   REMOVE_SAVED_REPORT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which to remove user saved interactive report information.</td>
</tr>
<tr>
<td>p_report_id</td>
<td>The ID of the saved user interactive report to be removed.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the REMOVE_SAVED_REPORT procedure to remove user saved interactive report with the ID 123 for the application with an ID of 100.

BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(100,123);
END;

17.13 REMOVE_SAVED_REPORTS Procedure

The REMOVE_SAVED_REPORTS procedure removes all user saved interactive report settings for a particular application or for the entire instance.

Syntax

APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(
    p_application_id    IN NUMBER DEFAULT NULL);

Parameters

Table 17-11   REMOVE_SAVED_REPORTS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which to remove user saved interactive report information. If this parameter is left null, all user saved interactive reports for the entire instance is removed.</td>
</tr>
</tbody>
</table>
Example

The following example demonstrates how to use the `REMOVE_SAVED_REPORTS` procedure to remove user saved interactive report information for the application with an ID of 100.

```sql
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(100);
END;
```

17.14 REMOVE_SCHEMA Procedure

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

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The name of the workspace from which the schema mapping is removed.</td>
</tr>
<tr>
<td>p_schema</td>
<td>The schema to remove from the schema to workspace mapping.</td>
</tr>
</tbody>
</table>

**Example**

The following example demonstrates how to use the `REMOVE_SCHEMA` procedure to remove the schema named Frank from the MY_WORKSPACE workspace to schema mapping.

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

17.15 REMOVE_SCHEMA_EXCEPTION Procedure

This procedure removes an exception that allows the assignment of a restricted schema to a given workspace.

**Syntax**

```sql
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA_EXCEPTION (  
    p_schema    in varchar2,  
    p_workspace in varchar2 );
```
Parameter

Table 17-13  REMOVE_SCHEMA_EXCEPTION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_schema</td>
<td>The schema.</td>
</tr>
<tr>
<td>p_workspace</td>
<td>The workspace.</td>
</tr>
</tbody>
</table>

Example

This example removes the exception that allows the assignment of schema HR to workspace HR_WORKSPACE.

begin
    apex_instance_admin.remove_schema_exception (
        p_schema     => 'HR',
        p_workspace  => 'HR_WORKSPACE');
    commit;
end;

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure"
- "RESTRICT_SCHEMA Procedure"
- "UNRESTRICT_SCHEMA Procedure"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure"

17.16 REMOVE_SCHEMA_EXCEPTIONS Procedure

This procedure removes all exceptions that allow the assignment of a given schema to workspaces.

Syntax

APEX_INSTANCE_ADMIN.REMOVE_SCHEMA_EXCEPTIONS ( 
    p_schema in varchar2 );
**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_schema</td>
<td>The schema.</td>
</tr>
</tbody>
</table>

**Example**

This example removes all exceptions that allow the assignment of the HR schema to workspaces.

```
begin
    apex_instance_admin.remove_schema_exceptions (
        p_schema => 'HR' );
    commit;
end;
```

**See Also:**
- "CREATE_SCHEMA_EXCEPTION Procedure"
- "RESTRICT_SCHEMA Procedure"
- "UNRESTRICT_SCHEMA Procedure"
- "REMOVE_SCHEMA_EXCEPTION Procedure"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure"

### 17.17 REMOVE_SUBSCRIPTION Procedure

The **REMOVE_SUBSCRIPTION** procedure removes a specific interactive report subscription.

**Syntax**

```
APEX_INSTANCE_ADMIN.REMOVE_SUBSCRIPTION(
    p_subscription_id IN NUMBER);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_subscription_id</td>
<td>The ID of the interactive report subscription to be removed.</td>
</tr>
</tbody>
</table>
Example

The following example demonstrates how to use the REMOVE_SUBSCRIPTION procedure to remove interactive report subscription with the ID 12345. Use of APEX_APPLICATION_PAGE_IR_SUB view can help identifying the subscription ID to remove.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SUBSCRIPTION (
        p_subscription_id => 12345);
END;
```

17.18 REMOVE_WORKSPACE Procedure

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The name of the workspace to be removed.</td>
</tr>
<tr>
<td>p_drop_users</td>
<td>'Y' to drop the database user associated with the workspace. The default is 'N'.</td>
</tr>
<tr>
<td>p_drop_tablespaces</td>
<td>'Y' to drop the tablespace associated with the database user associated with the workspace. The default is 'N'.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the REMOVE_WORKSPACE procedure to remove an existing workspace named MY_WORKSPACE, along with the associated database users and tablespace.

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

17.19 REMOVE_WORKSPACE_EXCEPTIONS Procedure

This procedure removes all exceptions that allow the assignment of restricted schemas to given workspace.
Syntax

APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE_EXCEPTIONS ( p_workspace in varchar2 );

Parameter

Table 17-17 REMOVE_WORKSPACE_EXCEPTIONS Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The workspace.</td>
</tr>
</tbody>
</table>

Example

This example removes all exceptions that allow the assignment of restricted schemas to HR_WORKSPACE.

begin    apex_instance_admin.remove_schema_exceptions ( p_workspace => 'HR_WORKSPACE' );    commit;end;

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure"
- "RESTRICT_SCHEMA Procedure"
- "UNRESTRICT_SCHEMA Procedure"
- "REMOVE_SCHEMA_EXCEPTION Procedure"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure"

17.20 RESERVE_WORKSPACE_APP_IDS Procedure

This procedure permanently reserves the IDs of websheet and database applications in a given workspace. Even if the workspace and its applications get removed, developers can not create other applications with one of these IDs.

Syntax

APEX_INSTANCE_ADMIN.RESERVE_WORKSPACE_APP_IDS ( p_workspace_id IN NUMBER );
Parameters

Table 17-18 RESERVE_WORKSPACE_APP_IDS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace_id</td>
<td>The unique ID of the workspace.</td>
</tr>
</tbody>
</table>

Example

This example demonstrates setting up two separate Application Express instances where the application IDs are limited to within a specific range. At a later point, a workspace and all of its applications are moved from instance 1 to instance 2. For the workspace that is moved, the developer reserves all of its application IDs to ensure that no applications with the same IDs are created on instance 1.

1. After setting up Application Express instance 1, ensure that application IDs are between 100000 and 199999.

   ```
   begin
   apex_instance_admin.set_parameter('APPLICATION_ID_MIN', 100000);
   apex_instance_admin.set_parameter('APPLICATION_ID_MAX', 199999);
   end;
   ```

2. After setting up Application Express instance 2, ensure that application IDs are between 200000 and 299999.

   ```
   begin
   apex_instance_admin.set_parameter('APPLICATION_ID_MIN', 200000);
   apex_instance_admin.set_parameter('APPLICATION_ID_MAX', 299999);
   end;
   ```

3. Later, the operations team decides that workspace MY_WORKSPACE with ID 1234567890 should be moved from instance 1 to instance 2. The required steps are:
   a. Export the workspace, applications and data on instance 1 (not shown here).
   b. Ensure that no other application on instance 1 can reuse application IDs of this workspace.

      ```
      begin
      apex_instance_admin.reserve_workspace_app_ids(1234567890);
      end;
      ```

   c. Drop workspace, accompanying data and users on instance 1.

      ```
      begin
      apex_instance_admin.remove_workspace('MY_WORKSPACE');
      end;
      ```

   d. Import the workspace, applications and data on instance 2 (not shown here).
17.21 RESTRICT_SCHEMA Procedure

This procedure revokes the privilege to assign a schema to workspaces.

Syntax

APEX_INSTANCE_ADMIN.RESTRICT_SCHEMA (p_schema in varchar2);

Parameter

Table 17-19 RESTRICT_SCHEMA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_schema</td>
<td>The schema.</td>
</tr>
</tbody>
</table>

Example

This example revokes the privilege to assign schema HR to workspaces.

begin
    apex_instance_admin.restrict_schema(p_schema => 'HR');
    commit;
end;

See Also:

- "FREE_WORKSPACE_APP_IDS Procedure"  
- "CREATE_SCHEMA_EXCEPTION Procedure"
- "UNRESTRICT_SCHEMA Procedure"
- "REMOVE_SCHEMA_EXCEPTION Procedure"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure"

17.22 SET_LOG_SWITCH_INTERVAL Procedure

Set the log switch interval for each of the logs maintained by Application Express.
Syntax

APEX_INSTANCE_ADMIN.SET_LOG_SWITCH_INTERVAL(
    p_log_name              IN VARCHAR2,
    p_log_switch_after_days IN NUMBER );

Parameters

Table 17-20  SET_LOG_SWITCH_INTERVAL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_log_name</td>
<td>Specifies the name of the log. Valid values include ACCESS, ACTIVITY, CLICKTHRU, and DEBUG.</td>
</tr>
<tr>
<td>p_log_switch_after_days</td>
<td>This interval must be a positive integer between 1 and 180.</td>
</tr>
</tbody>
</table>

Example

This example sets the log switch interval for the ACTIVITY log to 30 days.

begin
    apex_instance_admin.set_log_switch_interval( p_log_name => 'ACTIVITY',
                                             p_log_switch_after_days => 30 );
    commit;
end;

17.23 SET_WORKSPACE_PARAMETER

The SET_WORKSPACE_PARAMETER procedure sets the designated workspace parameter.

Syntax

SET_WORKSPACE_PARAMETER(
    p_workspace     IN VARCHAR2,
    p_parameter     IN VARCHAR2,
    p_value         IN VARCHAR2 );

Parameters

Table 17-21  SET_WORKSPACE_PARAMETER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The name of the workspace to which you are setting the workspace parameter.</td>
</tr>
</tbody>
</table>
Table 17-21  (Cont.) SET_WORKSPACE_PARAMETER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_parameter</td>
<td>The parameter name which overrides the instance parameter value of the same for this workspace. Parameter names include:</td>
</tr>
<tr>
<td></td>
<td>• ALLOW_HOSTNAMES</td>
</tr>
<tr>
<td></td>
<td>• MAX_SESSION_IDLE_SEC</td>
</tr>
<tr>
<td></td>
<td>• MAX_SESSION_LENGTH_SEC</td>
</tr>
<tr>
<td></td>
<td>• QOS_MAX_WORKSPACE_REQUESTS</td>
</tr>
<tr>
<td></td>
<td>• QOS_MAX_SESSION_REQUESTS</td>
</tr>
<tr>
<td></td>
<td>• QOS_MAX_SESSION_KILL_TIMEOUT</td>
</tr>
<tr>
<td></td>
<td>• RM_CONSUMER_GROUP</td>
</tr>
<tr>
<td></td>
<td>• WORKSPACE_EMAIL_MAXIMUM</td>
</tr>
<tr>
<td></td>
<td>• WORKSPACE_MAX_FILE_BYTES</td>
</tr>
<tr>
<td>p_value</td>
<td>The parameter value.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the `set_workspace_parameter` procedure to restrict URLs for accessing applications in the HR workspace that have hr.example.com in the hostname or domain name.

```
BEGIN
   apex_instance_admin.set_workspace_parameter (
      p_workspace  => 'HR',
      p_parameter  => 'ALLOW_HOSTNAMES',
      p_value      => 'hr.example.com');
END;
```

17.24 SET_PARAMETER Procedure

The `SET_PARAMETER` procedure sets a parameter used in administering a runtime environment. You must issue a commit for the parameter change to take affect.

Syntax

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

Parameters

Table 17-22  SET_PARAMETER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_parameter</td>
<td>The instance parameter to be set.</td>
</tr>
<tr>
<td>p_value</td>
<td>The value of the parameter.</td>
</tr>
</tbody>
</table>

See "Available Parameter Values".
Example

The following example demonstrates how to use the SET_PARAMETER procedure to set the SMTP_HOST_ADDRESS parameter for an Oracle Application Express instance.

BEGIN
    APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_HOST_ADDRESS',
    'mail.example.com');
    COMMIT;
END;

17.25 SET_WORKSPACE_CONSUMER_GROUP Procedure

The SET_WORKSPACE_CONSUMER_GROUP procedure sets a Resource Manager Consumer Group to a workspace.

Syntax

set_workspace_consumer_group(
    p_workspace in varchar2,
    p_rm_consumer_group in varchar2);

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>This is the name of the workspace for which the resource consumer group is to be set.</td>
</tr>
<tr>
<td>p_rm_consumer_group</td>
<td>The parameter P_RM_CONSUMER_GROUP is the Oracle Database Resource Manager Consumer Group name. The consumer group does not have to exist at the time this procedure is invoked. But if the Resource Manager Consumer Group is set for a workspace and the consumer group does not exist, then an error will be raised when anyone attempts to login to this workspace or execute any application in the workspace. If the value of P_RM_CONSUMER_GROUP is null, then the Resource Manager consumer group associated with the specified workspace is cleared.</td>
</tr>
</tbody>
</table>

Example

The following example sets the workspace to the Resource Manager consumer group "CUSTOM_GROUP1":

begin
    apex_instance_admin.set_workspace_consumer_group(
        p_workspace => 'MY_WORKSPACE',
        p_rm_consumer_group => 'CUSTOM_GROUP1');
end;
17.26 TRUNCATE_LOG Procedure

The TRUNCATE_LOG procedure truncates the log entries specified by the input parameter.

Syntax

APEX_INSTANCE_ADMIN.TRUNCATE_LOG(
    p_log     IN VARCHAR2);

Parameters

Table 17-24   TRUNCATE_LOG Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_log</td>
<td>This parameter can have one of the following values: ACTIVITY - removes all entries that record page access. USER_ACCESS - removes all entries that record user login. MAIL - removes all entries that record mail sent. CLICKS - removes all entries that record clicks tracked to external sites. LOCK_INSTALL_SCRIPT - removes all entries that record developer locking of supporting objects script. LOCK_PAGE - removes all entries that record developer locking of pages. WORKSPACE_HIST - removes all entries that record daily workspace summary. PURGE - removes all entries that record automatic workspace purge activity. FILE - removes all entries that record automatic file purge activity. SCRIPT - removes all entries that record results of SQL scripts executed in SQL Workshop. SQL - removes all entries that record the history of commands executed in SQL Workshop SQL Commands</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the TRUNCATE_LOG procedure to remove all log entries that record access to Application Express application pages.

BEGIN
    APEX_INSTANCE_ADMIN.TRUNCATE_LOG('ACTIVITY');
END;
17.27 UNRESTRICT_SCHEMA Procedure

This procedure re-grants the privilege to assign a schema to workspaces, if it has been revoked before.

Syntax

APEX_INSTANCE_ADMIN.UNRESTRICT_SCHEMA ( p_schema in varchar2 );

Parameter

Table 17-25 RESTRICT_SCHEMA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_schema</td>
<td>The schema.</td>
</tr>
</tbody>
</table>

Example

This example re-grants the privilege to assign schema HR to workspaces.

begin
    apex_instance_admin.unrestrict_schema(p_schema => 'HR');
    commit;
end;

See Also:

- "CREATE_SCHEMA_EXCEPTION Procedure"
- "RESTRICT_SCHEMA Procedure"
- "REMOVE_SCHEMA_EXCEPTION Procedure"
- "REMOVE_SCHEMA_EXCEPTIONS Procedure"
- "REMOVE_WORKSPACE_EXCEPTIONS Procedure"
The APEX_IR package provides utilities you can use when programming in the Oracle Application Express environment related to interactive reports. You can use the APEX_IR package to get an interactive report runtime query based on local and remote data source, add filters, reset or clear report settings, delete saved reports and manage subscriptions.

- ADD_FILTER Procedure Signature 1
- ADD_FILTER Procedure Signature 2
- CHANGE_SUBSCRIPTION_EMAIL Procedure
- CHANGE_REPORT_OWNER Procedure
- CHANGE_SUBSCRIPTION_EMAIL Procedure
- CHANGE_SUBSCRIPTION_LANG Procedure
- CLEAR_REPORT Procedure Signature 1
- CLEAR_REPORT Procedure Signature 2
- DELETE_REPORT Procedure
- DELETE_SUBSCRIPTION Procedure
- GET_LAST_VIEWED_REPORT_ID Function
- GET_REPORT Function
- RESET_REPORT Procedure Signature 1
- RESET_REPORT Procedure Signature 2

### 18.1 ADD_FILTER Procedure Signature 1

This procedure creates a filter on an interactive report using a report ID.

---

**Note:**

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.
Syntax

APEX_IR.ADD_FILTER(
    p_page_id       IN NUMBER,
    p_region_id     IN NUMBER,
    p_report_column IN VARCHAR2,
    p_filter_value  IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_report_id     IN NUMBER DEFAULT NULL);

Parameters

Table 18-1  ADD_FILTER Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region (ID).</td>
</tr>
<tr>
<td>p_report_column</td>
<td>Name of the report SQL column, or column alias, to be filtered.</td>
</tr>
<tr>
<td>p_filter_value</td>
<td>The filter value. This value is not used for N and NN.</td>
</tr>
<tr>
<td>p_operator_abbr</td>
<td>Filter type. Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td>EQ = Equals</td>
</tr>
<tr>
<td></td>
<td>NEQ = Not Equals</td>
</tr>
<tr>
<td></td>
<td>LT = Less than</td>
</tr>
<tr>
<td></td>
<td>LTE = Less then or equal to</td>
</tr>
<tr>
<td></td>
<td>GT = Greater Than</td>
</tr>
<tr>
<td></td>
<td>GTE = Greater than or equal to</td>
</tr>
<tr>
<td></td>
<td>LIKE = SQL Like operator</td>
</tr>
<tr>
<td></td>
<td>NLIKE = Not Like</td>
</tr>
<tr>
<td></td>
<td>N = Null</td>
</tr>
<tr>
<td></td>
<td>NN = Not Null</td>
</tr>
<tr>
<td></td>
<td>C = Contains</td>
</tr>
<tr>
<td></td>
<td>NC = Not Contains</td>
</tr>
<tr>
<td></td>
<td>IN = SQL In Operator</td>
</tr>
<tr>
<td></td>
<td>NIN = SQL Not In Operator</td>
</tr>
<tr>
<td>p_report_id</td>
<td>The saved report ID within the current application page. If p_report_id is null, it adds the filter to the last viewed report settings.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the ADD_FILTER procedure to filter the interactive report with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application with DEPTNO equals 30.

BEGIN
    APEX_IR.ADD_FILTER(
        p_page_id       => 1,
                p_region_id     => 2505704029884282,
                p_report_column => 'DEPTNO',
                p_filter_value  => '30',
                p_operator_abbr IN VARCHAR2 DEFAULT NULL,
                p_report_id     => NULL);
END;
18.2 ADD_FILTER Procedure Signature 2

This procedure creates a filter on an interactive report using a report alias.

Note:
The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

APEX_IR.ADD_FILTER(
    p_page_id       IN NUMBER,
    p_region_id     IN NUMBER,
    p_report_column IN VARCHAR2,
    p_filter_value  IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_report_alias  IN VARCHAR2 DEFAULT NULL);

Parameters

Table 18-2  ADD_FILTER Procedure Signature 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region (ID).</td>
</tr>
<tr>
<td>p_report_column</td>
<td>Name of the report SQL column, or column alias, to be filtered.</td>
</tr>
<tr>
<td>p_filter_value</td>
<td>This is the filter value. This value is not used for N and NN.</td>
</tr>
</tbody>
</table>
Table 18-2  (Cont.) ADD_FILTER Procedure Signature 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_operator_abbr</td>
<td>Filter type. Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td>EQ = Equals</td>
</tr>
<tr>
<td></td>
<td>NEQ = Not Equals</td>
</tr>
<tr>
<td></td>
<td>LT = Less than</td>
</tr>
<tr>
<td></td>
<td>LTE = Less then or equal to</td>
</tr>
<tr>
<td></td>
<td>GT = Greater Than</td>
</tr>
<tr>
<td></td>
<td>GTE = Greater than or equal to</td>
</tr>
<tr>
<td></td>
<td>LIKE = SQL Like operator</td>
</tr>
<tr>
<td></td>
<td>NLIKE = Not Like operator</td>
</tr>
<tr>
<td></td>
<td>N = Null</td>
</tr>
<tr>
<td></td>
<td>NN = Not Null</td>
</tr>
<tr>
<td></td>
<td>C = Contains</td>
</tr>
<tr>
<td></td>
<td>NC = Not Contains</td>
</tr>
<tr>
<td></td>
<td>IN = SQL In Operator</td>
</tr>
<tr>
<td></td>
<td>NIN = SQL Not In Operator</td>
</tr>
<tr>
<td>p_report_alias</td>
<td>The saved report alias within the current application page. If p_report_alias is null, it adds filter to the last viewed report settings.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the ADD_FILTER procedure to filter an interactive report with a report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application with DEPTNO equals 30.

BEGIN
    APEX_IR.ADD_FILTER(
        p_page_id       => 1,
        p_region_id     => 2505704029884282,
        p_report_column => 'DEPTNO',
        p_filter_value  => '30',
        p_operator_abbr => 'EQ',
        p_report_alias  => 'CATEGORY_REPORT');
END;

18.3 CHANGE_SUBSCRIPTION_EMAIL Procedure

This procedure changes interactive report subscriptions email address. When an email is sent out, the subscription sends message to the defined email address.

Syntax

APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (p_subscription_id IN NUMBER,
                                      p_email_address   IN VARCHAR2);
Parameters

Table 18-3  CHANGE_SUBSCRIPTION_EMAIL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_subscription_id</td>
<td>Subscription ID to change the email address within the current workspace.</td>
</tr>
<tr>
<td>p_email_address</td>
<td>The new email address to change to. The email address needs to be a valid email syntax and cannot be set to null.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use CHANGE_SUBSCRIPTION_EMAIL procedure to change the email address to some.user@somecompany.com for the interactive report subscription 956136850459718525.

BEGIN
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL ( 
  p_subscription_id => 956136850459718525,
  p_email_address => 'some.user@somecompany.com');
END;

18.4 CHANGE_REPORT_OWNER Procedure

This procedure changes the owner of a saved interactive report using a report ID. This procedure cannot change the owner of default interactive reports.

Syntax

APEX_IR.CHANGE_REPORT_OWNER ( 
  p_report_id in number,
  p_old_owner in varchar2,
  p_new_owner in varchar2);

Parameters

Table 18-4  CHANGE_REPORT_OWNER Procedure

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_report_id</td>
<td>The saved report ID within the current application page.</td>
</tr>
<tr>
<td>p_old_owner</td>
<td>The previous owner name to change from (case sensitive). The owner needs to be a valid login user accessing the report.</td>
</tr>
<tr>
<td>p_new_owner</td>
<td>The new owner name to change to (case sensitive). The owner must be a valid login user accessing the report.</td>
</tr>
</tbody>
</table>
Example

This example shows how to use CHANGE_REPORT_OWNER procedure to change the old owner name of JOHN to the new owner name of JOHN.DOE for a saved report. The saved report has a report ID of 1235704029884282.

```
BEGIN
APEX_IR.CHANGE_REPORT_OWNER (  
    p_report_id => 1235704029884282,  
    p_old_owner => 'JOHN',  
    p_new_owner => 'JOHN.DOE');
END;
```

18.5 CHANGE_SUBSCRIPTION_EMAIL Procedure

This procedure changes interactive report subscriptions email address. When an email is sent out, the subscription sends message to the defined email address.

Syntax

```
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (  
    p_subscription_id IN NUMBER,  
    p_email_address   IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Table 18-5</th>
<th>CHANGE_SUBSCRIPTION_EMAIL Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>p_subscription_id</td>
<td>Subscription ID to change the email address within the current workspace.</td>
</tr>
<tr>
<td>p_email_address</td>
<td>The new email address to change to. The email address needs to be a valid email syntax and cannot be set to null.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use CHANGE_SUBSCRIPTION_EMAIL procedure to change the email address to some.user@somecompany.com for the interactive report subscription 956136850459718525.

```
BEGIN
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (  
    p_subscription_id => 956136850459718525,  
    p_email_address => 'some.user@somecompany.com');
END;
```

18.6 CHANGE_SUBSCRIPTION_LANG Procedure

This procedure changes the interactive report subscription language.
Syntax

APEX_IR.CHANGE_SUBSCRIPTION_LANG(
    p_subscription_id IN NUMBER,
    p_language        IN VARCHAR2);

Parameters

Table 18-6  CHANGE_SUBSCRIPTION_LANG Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_subscription_id</td>
<td>Subscription ID to change the language within the current workspace.</td>
</tr>
<tr>
<td>p_language</td>
<td>This is an IANA language code. Some examples include: en, de, de-at, zh-cn, and pt-br.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the CHANGE_SUBSCRIPTION_LANG procedure to change the subscription with the ID of 567890123 to German in the current workspace.

```
BEGIN
    APEX_IR.CHANGE_SUBSCRIPTION_LANG(
        p_subscription_id => 567890123,
        p_language        => 'de');
END;
```

18.7 CLEAR_REPORT Procedure Signature 1

This procedure clears report settings using the report ID.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

APEX_IR.CLEAR_REPORT(
    p_page_id   IN NUMBER,
Parameters

Table 18-7  CLEAR_REPORT Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region (ID).</td>
</tr>
<tr>
<td>p_report_id</td>
<td>The saved report ID within the current application page. If p_report_id is null, it clears the last viewed report settings.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the CLEAR_REPORT procedure to clear interactive report settings with a report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

BEGIN
    APEX_IR.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_id    => 880629800374638220);
END;

18.8 CLEAR_REPORT Procedure Signature 2

This procedure clears report settings using report alias.

**Note:**

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

APEX_IR.CLEAR_REPORT(
    p_page_id     IN NUMBER,
    p_region_id   IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
Parameters

Table 18-8 CLEAR_REPORT Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region (ID).</td>
</tr>
<tr>
<td>p_report_alias</td>
<td>The saved report alias within the current application page. If p_report_alias is null, it clears the last viewed report settings.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the CLEAR_REPORT procedure to clear interactive report settings with report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application.

```
BEGIN
  APEX_IR.CLEAR_REPORT(
    p_page_id      => 1,
    p_region_id    => 2505704029884282,
    p_report_alias => 'CATEGORY_REPORT');
END;
```

18.9 DELETE_REPORT Procedure

This procedure deletes saved interactive reports. It deletes a specific saved report in the current logged in workspace and application.

Syntax

```
APEX_IR.DELETE_REPORT(
  p_report_id IN NUMBER);
```

Parameters

Table 18-9 DELETE_REPORT Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_report_id</td>
<td>Report ID to delete within the current Application Express application.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the DELETE_REPORT procedure to delete the saved interactive report with ID of 880629800374638220 in the current application.

```
BEGIN
  APEX_IR.DELETE_REPORT (
```

18-9
18.10 DELETE_SUBSCRIPTION Procedure

This procedure deletes interactive report subscriptions.

Syntax

APEX_IR.DELETE_SUBSCRIPTION(
    p_subscription_id IN NUMBER);

Parameters

Table 18-10  DELETE_SUBSCRIPTION Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_subscription_id</td>
<td>Subscription ID to delete within the current workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the DELETE_SUBSCRIPTION procedure to delete the subscription with ID of 567890123 in the current workspace.

BEGIN
    APEX_IR.DELETE_SUBSCRIPTION(
        p_subscription_id => 567890123);
END;

18.11 GET_LAST_VIEWED_REPORT_ID Function

This function returns the last viewed base report ID of the specified page and region.

Syntax

APEX_IR.GET_LAST_VIEWED_REPORT_ID(
    p_page_id   IN NUMBER,
    p_region_id IN NUMBER);

Parameters

Table 18-11  GET_LAST_VIEWED_REPORT_ID Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region ID.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the `GET_LAST_VIEWED_REPORT_ID` function to retrieve the last viewed report ID in page 1, region 2505704029884282 of the current application.

```sql
DECLARE
    l_report_id number;
BEGIN
    l_report_id := APEX_IR.GET_LAST_VIEWED_REPORT_ID (p_page_id => 1,
                                                        p_region_id => 2505704029884282);
END;
```

18.12 GET_REPORT Function

This function returns an interactive report runtime query.

Syntax

```sql
APEX_IR.GET_REPORT(
    p_page_id    IN NUMBER,
    p_region_id IN NUMBER,
    p_report_id IN NUMBER DEFAULT NULL,
    p_view       IN VARCHAR2 DEFAULT APEX_IR.C_VIEW_REPORT);
```

Parameters

Table 18-12  GET_REPORT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region ID.</td>
</tr>
<tr>
<td>p_report_id</td>
<td>The saved report ID within the current application page. If p_report_id is null, it gets last viewed report query.</td>
</tr>
<tr>
<td>p_view</td>
<td>The view type available for the report. The values can be APEX_IR.C_VIEW_REPORT, APEX_IR.C_VIEW_GROUPBY or APEX_IR.C_VIEW_PIVOT. If p_view is null, it gets the view currently used by the report. If p_view passed which doesn't exist for the current report, an error is raised.</td>
</tr>
</tbody>
</table>

Example 1

The following example shows how to use the `GET_REPORT` function to retrieve the runtime report query with bind variable information with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```sql
DECLARE
    l_report apex_ir.t_report;
    l_query varchar2(32767);
```
BEGIN
   l_report := APEX_IR.GET_REPORT (
      p_page_id => 1,
      p_region_id => 2505704029884282,
      p_report_id => 880629800374638220);
   l_query := l_report.sql_query;
   sys.htp.p('Statement = '||l_report.sql_query);
   for i in 1..l_report.binds.count
      loop
      sys.htp.p(i||'.'||l_report.binds(i).name||' = '||l_report.binds(i).value);
   end loop;
END;

Example 2
The following example shows how to use the GET_REPORT function to retrieve Group By view query defined in the current report page with region 2505704029884282.

DECLARE
   l_report APEX_IR.T_REPORT;
BEGIN
   l_report := APEX_IR.GET_REPORT (
      p_page_id        => :APP_PAGE_ID,
      p_region_id      => 2505704029884282,
      p_view           => APEX_IR.C_VIEW_GROUPBY );
   sys.htp.p( 'Statement = '||l_report.sql_query );
END;

18.13 RESET_REPORT Procedure Signature 1

This procedure resets report settings to the developer defined default settings using the report ID.

Note:
The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

APEX_IR.RESET_REPORT(
   p_page_id IN NUMBER,  
   p_region_id IN NUMBER,  
   p_report_id IN NUMBER DEFAULT NULL);
Parameters

Table 18-13  RESET_REPORT Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region ID.</td>
</tr>
<tr>
<td>p_report_id</td>
<td>The saved report ID within the current application page. If p_report_id is null, it resets</td>
</tr>
<tr>
<td></td>
<td>the last viewed report settings.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the RESET_REPORT procedure signature 1 to reset interactive report settings with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.RESET_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_id    => 880629800374638220);
END;
```

18.14 RESET_REPORT Procedure Signature 2

This procedure resets report settings using the report alias.

Note:

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.RESET_REPORT(
    p_page_id IN NUMBER,
    p_region_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```
Parameters

Table 18-14  RESET_REPORT Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>The interactive report region ID.</td>
</tr>
<tr>
<td>p_report_alias</td>
<td>The saved report alias within the current application page. If p_report_alias is null, it resets the last viewed report settings.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the RESET_REPORT procedure to reset interactive report settings with a report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application.

BEGIN
APEX_IR.RESET_REPORT(
    p_page_id      => 1,
    p_region_id    => 2505704029884282,
    p_report_alias => 'CATEGORY_REPORT');
END;
19

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.

- CHECKBOX2 Function
- DATE_POPUP Function
- DATE_POPUP2 Function
- DISPLAY_AND_SAVE Function
- HIDDEN Function
- MD5_CHECKSUM Function
- MD5_HIDDEN Function
- POPUP_FROM_LOV Function
- POPUP_FROM_QUERY Function
- POPUPKEY_FROM_LOV Function
- POPUPKEY_FROM_QUERY Function
- RADIOGROUP Function
- SELECT_LIST Function
- SELECT_LIST_FROM_LOV Function
- SELECT_LIST_FROM_LOV_XL Function
- SELECT_LIST_FROM_QUERY Function
- SELECT_LIST_FROM_QUERY_XL Function
- SWITCH Function
- TEXT Function
- TEXTAREA Function
- TEXT_FROM_LOV Function
- TEXT_FROM_LOV_QUERY Function

19.1 CHECKBOX2 Function

This function creates check boxes.

Syntax

APEX_ITEM.CHECKBOX2 (p_idx NUMBER, p_value VARCHAR2 DEFAULT NULL, p_attributes VARCHAR2 DEFAULT NULL, ...
p_checked_values           IN    VARCHAR2 DEFAULT NULL,
p_checked_values_delimiter IN    VARCHAR2 DEFAULT ':',
p_item_id                  IN    VARCHAR2 DEFAULT NULL,
p_item_label               IN    VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;

Parameters

Table 19-1  CHECKBOX2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number that determines which APEX_APPLICATION global variable is used. Valid</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of a check box, hidden field, or input form item</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Controls the size of the text field</td>
</tr>
<tr>
<td>p_checked_values</td>
<td>Values to be checked by default</td>
</tr>
<tr>
<td>p_checked_values_delimiter</td>
<td>Delimits the values in the previous parameter, p_checked_values</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item</td>
</tr>
</tbody>
</table>

Examples of Default Check Box Behavior

The following example demonstrates how to create a selected check box for each employee in the emp table.

```sql
SELECT APEX_ITEM.CHECKBOX2(1,empno,'CHECKED') "Select",
       ename, job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to have all check boxes for employees display without being selected.

```sql
SELECT APEX_ITEM.CHECKBOX2(1,empno) "Select",
       ename, job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to select the check boxes for employees who work in department 10.

```sql
SELECT APEX_ITEM.CHECKBOX2(1,empno,DECODE(deptno,10,'CHECKED',NULL)) "Select",
       ename, job
FROM   emp
ORDER BY 1
```
The next example demonstrates how to select the check boxes for employees who work in department 10 or department 20.

```sql
SELECT APEX_ITEM.CHECKBOX2(1,deptno,NULL,'10:20',':') "Select",
       ename, job
FROM emp
ORDER BY 1
```

### Creating an On-Submit Process

If you are using check boxes in your application, you might need to create an On Submit process to perform a specific type of action on the selected rows. For example, you could have a Delete button that uses the following logic:

```sql
SELECT APEX_ITEM.CHECKBOX2(1,empno) "Select",
       ename, job
FROM emp
ORDER by 1
```

Consider the following sample on-submit process:

```sql
FOR I in 1..APEX_APPLICATION.G_F01.COUNT LOOP
    DELETE FROM emp WHERE empno = to_number(APEX_APPLICATION.G_F01(i));
END LOOP;
```

The following example demonstrates how to create unselected checkboxes for each employee in the emp table, with a unique ID. This is useful for referencing records from within JavaScript code:

```sql
SELECT APEX_ITEM.CHECKBOX2(1,empno,NULL,NULL,NULL,'f01_#ROWNUM#') "Select",
       ename, job
FROM emp
ORDER BY 1
```

### 19.2 DATE_POPUP Function

Use this function with forms that include date fields. The `DATE_POPUP` function dynamically generates a date field that has a popup calendar button.

#### Syntax

```sql
APEX_ITEM.DATE_POPUP(
    p_idx                       IN    NUMBER,
    p_row                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT NULL,
    p_date_format               IN    DATE DEFAULT 'DD-MON-YYYY',
    p_size                      IN    NUMBER DEFAULT 20,
    p_maxlength                 IN    NUMBER DEFAULT 2000,
    p_attributes                IN    VARCHAR2 DEFAULT NULL,
    p_item_id                   IN    VARCHAR2 DEFAULT NULL,
)
```
Parameters

Table 19-2  DATE_POPUP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02.</td>
</tr>
<tr>
<td>p_row</td>
<td>This parameter is deprecated. Anything specified for this value is ignored.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of a field item.</td>
</tr>
<tr>
<td>p_date_format</td>
<td>Valid database date format.</td>
</tr>
<tr>
<td>p_size</td>
<td>Controls HTML tag attributes (such as disabled).</td>
</tr>
<tr>
<td>p_maxlength</td>
<td>Determines the maximum number of enterable characters. Becomes the maxlength attribute of the &lt;input&gt; HTML tag.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use APEX_ITEM.DATE_POPUP to create popup calendar buttons for the hiredate column.

```sql
SELECT empno,
    APEX_ITEM.HIDDEN(1,empno)||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hd,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

See Also:

Oracle Database SQL Language Reference for information about the TO_CHAR or TO_DATE functions.
19.3 DATE_POPUP2 Function

Use this function with forms that include date fields. The DATE_POPUP2 function dynamically generates a date field that has a jQuery based popup calendar with button.

Syntax

APEX_ITEM.DATE_POPUP2(
    p_idx                   in number,
    p_value                 in date     default null,
    p_date_format           in varchar2 default null,
    p_size                  in number   default 20,
    p_maxLength             in number   default 2000,
    p_attributes            in varchar2 default null,
    p_item_id               in varchar2 default null,
    p_item_label            in varchar2 default null,
    p_default_value         in varchar2 default null,
    p_max_value             in varchar2 default null,
    p_min_value             in varchar2 default null,
    p_show_on               in varchar2 default 'button',
    p_number_of_months      in varchar2 default null,
    p_navigation_list_for   in varchar2 default 'NONE',
    p_year_range            in varchar2 default null,
    p_validation_date       in varchar2 default null)
RETURN VARCHAR2;

Parameters

Table 19-3  DATE_POPUP2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of a field item</td>
</tr>
<tr>
<td>p_date_format</td>
<td>Valid database date format</td>
</tr>
<tr>
<td>p_size</td>
<td>Controls HTML tag attributes (such as disabled)</td>
</tr>
<tr>
<td>p_maxlength</td>
<td>Determines the maximum number of enterable characters. Becomes the maxlength attribute of the &lt;input&gt; HTML tag</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item</td>
</tr>
<tr>
<td>p_default_value</td>
<td>The default date which should be selected in DatePicker calendar popup</td>
</tr>
<tr>
<td>p_max_value</td>
<td>The Maximum date that can be selected from the datepicker</td>
</tr>
</tbody>
</table>
### Table 19-3  (Cont.) DATE_POPUP2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{\text{min_value}} )</td>
<td>The Minimum date that can be selected from the datepicker.</td>
</tr>
<tr>
<td>( p_{\text{show_on}} )</td>
<td>Determines when the datepicker displays, on button click or on focus of the item or both.</td>
</tr>
<tr>
<td>( p_{\text{number_of_months}} )</td>
<td>Determines number of months displayed. Value should be in array formats follows: ([\text{row, column}])</td>
</tr>
<tr>
<td>( p_{\text{navigation_list_for}} )</td>
<td>Determines if a select list is displayed for Changing Month, Year or Both. Possible values include: MONTH,YEAR,MONTH_AND_YEAR and default is null.</td>
</tr>
<tr>
<td>( p_{\text{year_range}} )</td>
<td>The range of years displayed in the year selection list.</td>
</tr>
<tr>
<td>( p_{\text{validation_date}} )</td>
<td>Used to store the Date value for the which date validation failed</td>
</tr>
</tbody>
</table>

### See Also:

*Oracle Database SQL Language Reference* for information about the `TO_CHAR` or `TO_DATE` functions

### 19.4 DISPLAY_AND_SAVE Function

Use this function to display an item as text, but save its value to session state.

**Syntax**

```sql
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 19-4  DISPLAY_AND_SAVE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{\text{idx}} )</td>
<td>Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02</td>
</tr>
<tr>
<td>( p_{\text{value}} )</td>
<td>Current value</td>
</tr>
<tr>
<td>( p_{\text{item_id}} )</td>
<td>HTML attribute ID for the (&lt;\text{span}&gt;) tag</td>
</tr>
</tbody>
</table>
Table 19-4  (Cont.) DISPLAY_AND_SAVE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the APEX_ITEM.DISPLAY_AND_SAVE function.

```sql
SELECT APEX_ITEM.DISPLAY_AND_SAVE(10, empno) c FROM emp
```

19.5 HIDDEN Function

This function dynamically generates hidden form items.

Syntax

APEX_ITEM.HIDDEN(
  p_idx IN NUMBER,
  p_value IN VARCHAR2 DEFAULT NULL,
  p_attributes IN VARCHAR2 DEFAULT NULL,
  p_item_id IN VARCHAR2 DEFAULT NULL,
  p_item_label IN VARCHAR2 DEFAULT NULL
) RETURN VARCHAR2;

Parameters

Table 19-5  HIDDEN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number to identify the item you want to generate. The number determines which G_FXX global is populated. See Also: &quot;APEX_APPLICATION&quot;</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of the hidden input form item</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item</td>
</tr>
</tbody>
</table>

Example

Typically, the primary key of a table is stored as a hidden column and used for subsequent update processing, for example:

```sql
SELECT empno,
  APEX_ITEM.HIDDEN(1, empno) ||
  APEX_ITEM.TEXT(2, ename) ename,
```
APEX_ITEM.TEXT(3, job) job,
APEX_ITEM.TEXT(4, 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.

### 19.6 MD5_CHECKSUM Function

Use this function for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces hidden form field(s) with a name attribute equal to ‘fcs’ and as value a MD5 checksum based on up to 50 inputs. APEX_ITEM.MD5_CHECKSUM also produces an MD5 checksum using Oracle database DBMS_CRYPTO:

```sql
UTL_RAW.CAST_TO_RAW(DBMS_CRYPTO.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network.

**Syntax**

```sql
APEX_ITEM.MD5_CHECKSUM(
    p_value01 IN VARCHAR2 DEFAULT NULL,
    p_value02 IN VARCHAR2 DEFAULT NULL,
    p_value03 IN VARCHAR2 DEFAULT NULL,
    ...,
    p_value50 IN VARCHAR2 DEFAULT NULL,
    p_col_sep IN VARCHAR2 DEFAULT '|',
    p_item_id IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```
Parameters

Table 19-6  MD5_CHECKSUM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value01</td>
<td>Fifty available inputs. If no parameters are supplied, the default to NULL</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>p_value50</td>
<td></td>
</tr>
<tr>
<td>p_col_sep</td>
<td>String used to separate p_value inputs. Defaults to the pipe symbol (</td>
</tr>
<tr>
<td>p_item_id</td>
<td>ID of the HTML form item</td>
</tr>
</tbody>
</table>

Example

This function generates hidden form elements with the name 'fcs'. The values can subsequently be accessed by using the APEX_APPLICATION.G_FCS array.

```sql
SELECT APEX_ITEM.MD5_CHECKSUM(ename, job, sal) md5_cks,
       ename, job, sal
FROM emp;
```

### 19.7 MD5_HIDDEN Function

Use this function for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces a hidden form field with a MD5 checksum as value which is based on up to 50 inputs. APEX_ITEM.MD5_HIDDEN also produces an MD5 checksum using Oracle database DBMS_CRYPTO:

```
UTL_RAW.CAST_TO_RAW(DBMS_CRYPTO.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network.

**Syntax**

```sql
APEX_ITEM.MD5_HIDDEN(
  p_idx       IN    NUMBER,
  p_value01   IN    VARCHAR2 DEFAULT NULL,
  p_value02   IN    VARCHAR2 DEFAULT NULL,
  p_value03   IN    VARCHAR2 DEFAULT NULL,
  ...
  p_value50  IN    VARCHAR2 DEFAULT NULL,
  p_col_sep  IN    VARCHAR2 DEFAULT ',',
  p_item_id  IN    VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```
Parameters

Table 19-7  MD5_HIDDEN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>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</td>
</tr>
<tr>
<td>p_value01</td>
<td>Fifty available inputs. Parameters not supplied default to NULL</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>p_value50</td>
<td></td>
</tr>
<tr>
<td>p_col_sep</td>
<td>String used to separate p_value inputs. Defaults to the pipe symbol (</td>
</tr>
<tr>
<td>p_item_id</td>
<td>ID of the HTML form item</td>
</tr>
</tbody>
</table>

Example

The p_idx parameter specifies the FXX form element to be generated. In the following example, 7 generates F07. Also note that an HTML hidden form element is generated.

```
SELECT APEX_ITEM.MD5_HIDDEN(7, ename, job, sal) md5_h, ename, job, sal
FROM emp
```

19.8 POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application shared list of values (LOV). Like other available functions in the APEX_ITEM package, POPUP_FROM_LOV function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUP_FROM_LOV(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT 'false',
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'true',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```
Parameters

Table 19-8  POPUP_FROM_LOV Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Form element current value. This value should be one of the values in the p_lov_name parameter.</td>
</tr>
<tr>
<td>p_lov_name</td>
<td>Named LOV used for this popup.</td>
</tr>
<tr>
<td>p_width</td>
<td>Width of the text box.</td>
</tr>
<tr>
<td>p_max_length</td>
<td>Maximum number of characters that can be entered in the text box.</td>
</tr>
<tr>
<td>p_form_index</td>
<td>HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.</td>
</tr>
<tr>
<td>p_escape_html</td>
<td>Replacements for special characters that require an escaped equivalent: • &lt; for &lt; • &gt; for &gt; • &amp; for &amp; Range of values is YES and NO. If YES, special characters are escaped. This parameter is useful if you know your query returns illegal HTML.</td>
</tr>
<tr>
<td>p_max_elements</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional HTML attributes to use for the form item.</td>
</tr>
<tr>
<td>p_ok_to_query</td>
<td>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.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>ID attribute of the form element.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates a sample query the generates a popup from an LOV named DEPT_LOV.

SELECT APEX_ITEM.POPUP_FROM_LOV (1,deptno,'DEPT_LOV') dt FROM emp
19.9 POPUP_FROM_QUERY Function

This function generates an HTML popup select list from a query. Like other available functions in the APEX_ITEM package, the POPUP_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

APEX_ITEM.POPUP_FROM_QUERY(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;

Parameters

Table 19-9  POPUP_FROM_QUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Form element current value. This value should be one of the values in the p_lov_query parameter.</td>
</tr>
<tr>
<td>p_lov_query</td>
<td>SQL query that is expected to select two columns (a display column and a return column). For example: SELECT dname, deptno FROM dept</td>
</tr>
<tr>
<td>p_width</td>
<td>Width of the text box.</td>
</tr>
<tr>
<td>p_max_length</td>
<td>Maximum number of characters that can be entered in the text box.</td>
</tr>
<tr>
<td>p_form_index</td>
<td>HTML form on the page in which an item is contained. Defaults to 0 and rarely used.</td>
</tr>
<tr>
<td>p_escape_html</td>
<td>Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.</td>
</tr>
<tr>
<td>p_max_elements</td>
<td></td>
</tr>
<tr>
<td>p_attributes</td>
<td></td>
</tr>
<tr>
<td>p_ok_to_query</td>
<td></td>
</tr>
<tr>
<td>p_item_id</td>
<td></td>
</tr>
<tr>
<td>p_item_label</td>
<td></td>
</tr>
</tbody>
</table>
Table 19-9  (Cont.) POPUP_FROM_QUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_escape_html</td>
<td>Replacements for special characters that require an escaped equivalent.</td>
</tr>
<tr>
<td></td>
<td>•  &lt; for &lt;</td>
</tr>
<tr>
<td></td>
<td>•  &gt; for &gt;</td>
</tr>
<tr>
<td></td>
<td>•  &amp; for &amp;</td>
</tr>
<tr>
<td></td>
<td>Range of values is YES and NO. If YES, special characters are escaped. This parameter is useful if you know your query returns illegal HTML.</td>
</tr>
<tr>
<td>p_max_elements</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional HTML attributes to use for the form item.</td>
</tr>
<tr>
<td>p_ok_to_query</td>
<td>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.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>ID attribute of the form element.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates a sample query the generates a popup select list from the emp table.

```sql
SELECT APEX_ITEM.POPUP_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

19.10 POPUPKEY_FROM_LOV Function

This function generates a popup key select list from a shared list of values (LOV). Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_LOV function is designed to generate forms with F01 to F50 form array elements.

Syntax

```sql
APEX_ITEM.POPUPKEY_FROM_LOV(
    p_idx IN NUMBER,
    p_value IN VARCHAR2 DEFAULT NULL,
    p_lov_name IN VARCHAR2,
    p_width IN VARCHAR2 DEFAULT NULL,
    p_max_length IN VARCHAR2 DEFAULT NULL,
    p_form_index IN VARCHAR2 DEFAULT '0',
    p_escape_html IN VARCHAR2 DEFAULT NULL,
    p_max_elements IN VARCHAR2 DEFAULT NULL,
    p_attributes IN VARCHAR2 DEFAULT NULL,
    p_ok_to_query IN VARCHAR2 DEFAULT 'YES',
)```
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 19-10  POPUPKEY_FROM_LOV Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>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:</td>
</tr>
<tr>
<td>p_value</td>
<td>Indicates the current value. This value should be one of the values in the P_LOV_NAME parameter.</td>
</tr>
<tr>
<td>p_lov_name</td>
<td>Identifies a named LOV used for this popup.</td>
</tr>
<tr>
<td>p_width</td>
<td>Width of the text box.</td>
</tr>
<tr>
<td>p_max_length</td>
<td>Maximum number of characters that can be entered in the text box.</td>
</tr>
<tr>
<td>p_form_index</td>
<td>HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.</td>
</tr>
<tr>
<td>p_escape_html</td>
<td>Replacements for special characters that require an escaped equivalent.</td>
</tr>
<tr>
<td>p_max_elements</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional HTML attributes to use for the form item.</td>
</tr>
<tr>
<td>p_ok_to_query</td>
<td>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.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item</td>
</tr>
</tbody>
</table>
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

19.11 POPUPKEY_FROM_QUERY Function

This function generates a popup key select list from a SQL query. Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

APEX_ITEM.POPUPKEY_FROM_QUERY(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;

Parameters

Table 19-11  POPUPKEY_FROM_QUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>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</td>
</tr>
<tr>
<td>p_value</td>
<td>Form element current value. This value should be one of the values in the P_LOV_QUERY parameter.</td>
</tr>
<tr>
<td>p_lov_query</td>
<td>LOV query used for this popup.</td>
</tr>
<tr>
<td>p_width</td>
<td>Width of the text box.</td>
</tr>
<tr>
<td>p_max_length</td>
<td>Maximum number of characters that can be entered in the text box.</td>
</tr>
</tbody>
</table>
Table 19-11  (Cont.) POPUPKEY_FROM_QUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_form_index</td>
<td>HTML form on the page in which an item is contained. Defaults to 0 and rarely used.</td>
</tr>
<tr>
<td></td>
<td>Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.</td>
</tr>
<tr>
<td>p_escape_html</td>
<td>Replacements for special characters that require an escaped equivalent.</td>
</tr>
<tr>
<td></td>
<td>• &lt; for &lt;</td>
</tr>
<tr>
<td></td>
<td>• &gt; for &gt;</td>
</tr>
<tr>
<td></td>
<td>• &amp; for &amp;</td>
</tr>
<tr>
<td></td>
<td>This parameter is useful if you know your query returns illegal HTML.</td>
</tr>
<tr>
<td>p_max_elements</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional HTML attributes to use for the form item.</td>
</tr>
<tr>
<td>p_ok_to_query</td>
<td>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.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>ID attribute of the form element.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to generate a popup select list from a SQL query.

```sql
SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

19.12 RADIOGROUP Function

This function generates a radio group from a SQL query.

Syntax

```sql
APEX_ITEM.RADIOGROUP (p_idx IN NUMBER,
p_value IN VARCHAR2 DEFAULT NULL,
p_selected_value IN VARCHAR2 DEFAULT NULL,
p_display IN VARCHAR2 DEFAULT NULL,
p_attributes IN VARCHAR2 DEFAULT NULL,
p_onblur IN VARCHAR2 DEFAULT NULL,
```
### Parameters

**Table 19-12**  RADIOGROUP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example 1 creates F01 and 2 creates F02.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of the radio group.</td>
</tr>
<tr>
<td>p_selected_value</td>
<td>Value that should be selected.</td>
</tr>
<tr>
<td>p_display</td>
<td>Text to display next to the radio option.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_onblur</td>
<td>JavaScript to execute in the onBlur event.</td>
</tr>
<tr>
<td>p_onchange</td>
<td>JavaScript to execute in the onChange event.</td>
</tr>
<tr>
<td>p_onfocus</td>
<td>JavaScript to execute in the onFocus event.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

### Example

The following example demonstrates how to select department 20 from the emp table as a default in a radio group.

```sql
SELECT APEX_ITEM.RADIOGROUP (1, deptno, '20', dname) dt
FROM   dept
ORDER  BY 1
```

### 19.13 SELECT_LIST Function

This function dynamically generates a static select list. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

**Syntax**

APEX_ITEM.SELECT_LIST(
    p_idx IN NUMBER,
    p_value IN VARCHAR2 DEFAULT NULL,
    p_list_values IN VARCHAR2 DEFAULT NULL,
    p_attributes IN VARCHAR2 DEFAULT NULL,
    p_show_null IN VARCHAR2 DEFAULT 'NO',
    p_null_value IN VARCHAR2 DEFAULT '%NULL%'
) RETURN VARCHAR2;
p_null_text IN VARCHAR2 DEFAULT '%%',
p_item_id IN VARCHAR2 DEFAULT NULL,
p_item_label IN VARCHAR2 DEFAULT NULL,
p_show_extra IN VARCHAR2 DEFAULT 'YES')
RETURN VARCHAR2;

Parameters

Table 19-13  SELECT_LIST Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P IDX parameter is constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Current value. This value should be a value in the P_LIST_VALUES parameter.</td>
</tr>
<tr>
<td>p_list_values</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_show_null</td>
<td>Extra select option to enable the NULL selection. Range of values is YES and NO.</td>
</tr>
<tr>
<td>p_null_value</td>
<td>Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
<tr>
<td>p_show_extra</td>
<td>Shows the current value even if the value of p_value is not located in the select list.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates a static select list that displays Yes, returns Y, defaults to Y, and generates a F01 form item.

```
SELECT APEX_ITEM.SELECT_LIST(1,'Y','Yes;Y,No;N')yn
FROM emp
```

The following example demonstrates the use of APEX_ITEM.SELECT_LIST to generate a static select list where:

- A form array element F03 is generated (p_idx parameter).
- The initial value for each element is equal to the value for deptno for the row from emp (p_value parameter).
- The select list contains 4 options (p_list_values parameter).
- The text within the select list displays in red (p_attributes parameter).
• A null option is displayed (p_show_null) and this option displays -Select- as the text (p_null_text parameter).

• An HTML ID attribute is generated for each row, where #ROWNUM# is substituted for the current row rownum (p_item_id parameter). (So an ID of 'f03_4' is generated for row 4.)

• A HTML label element is generated for each row (p_item_label parameter).

• The current value for deptno is displayed, even if it is not contained with the list of values passed in the p_list_values parameter (p_show_extra parameter).

```
SELECT  empno "Employee ",
        ename "Name",
        APEX_ITEM.SELECT_LIST (p_idx     =>   3,
                               p_value   =>   deptno,
                               p_list_values =>   'ACCOUNTING;10,RESEARCH;20,SALES;
                                                       30,OPERATIONS;40',
                               p_attributes =>   'style="color:red;"',
                               p_show_null   =>   'YES',
                               p_null_value  =>   NULL,
                               p_null_text   =>   '-Select-',
                               p_item_id     =>   'f03_#ROWNUM#',
                               p_item_label  =>   'Label for f03_#ROWNUM#',
                               p_show_extra  =>   'YES') "Department"
FROM  emp;
```

19.14 SELECT_LIST_FROM_LOV Function

This function dynamically generates select lists from a shared list of values (LOV). Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements. This function is the same as SELECT_LIST_FROM_LOV, but its return value is CLOB. Use this function in SQL queries where you need to handle a column value longer than 4000 characters.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV (p_idx     IN   NUMBER,
                                 p_value   IN   VARCHAR2 DEFAULT NULL,
                                 p_lov     IN   VARCHAR2,
                                 p_attributes IN   VARCHAR2 DEFAULT NULL,
                                 p_show_null   IN   VARCHAR2 DEFAULT 'YES',
                                 p_null_value  IN   VARCHAR2 DEFAULT '%NULL%',
                                 p_null_text   IN   VARCHAR2 DEFAULT '%',
                                 p_item_id     IN   VARCHAR2 DEFAULT NULL,
                                 p_item_label  IN   VARCHAR2 DEFAULT NULL,
                                 p_show_extra  IN   VARCHAR2 DEFAULT 'YES')
RETURN VARCHAR2;
```
### Parameters

**Table 19-14 SELECT_LIST_FROM_LOV Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Current value. This value should be a value in the p_lov parameter.</td>
</tr>
<tr>
<td>p_lov</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_show_null</td>
<td>Extra select option to enable the NULL selection. Range of values is YES and NO.</td>
</tr>
<tr>
<td>p_null_value</td>
<td>Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;select&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
<tr>
<td>p_show_extra</td>
<td>Shows the current value even if the value of p_value is not located in the select list.</td>
</tr>
</tbody>
</table>

### Example

The following example demonstrates a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV(2, job, 'JOB_FLOW_LOV') job
FROM emp
```

### 19.15 SELECT_LIST_FROM_LOV_XL Function

This function dynamically generates very large select lists (greater than 32K) from a shared list of values (LOV). Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements. This function is the same as SELECT_LIST_FROM_LOV, but its return value is CLOB. Returned values will be limited to 32k.

**Syntax**

```
APEX_ITEM.SELECT_LIST_FROM_LOV_XL(
  p_idx           IN   NUMBER,
  p_value         IN   VARCHAR2 DEFAULT NULL,
  p_lov           IN   VARCHAR2,
  p_attributes    IN   VARCHAR2 DEFAULT NULL,
  p_show_null     IN   VARCHAR2 DEFAULT ©YES©,
  p_null_value    IN   VARCHAR2 DEFAULT ©%NULL%©,
)
```
Parameters

Table 19-15  SELECT_LIST_FROM_LOV_XL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Current value. This value should be a value in the p_lov parameter.</td>
</tr>
<tr>
<td>p_lov</td>
<td>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.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_show_null</td>
<td>Extra select option to enable the NULL selection. Range of values is YES and NO.</td>
</tr>
<tr>
<td>p_null_value</td>
<td>Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;select&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
<tr>
<td>p_show_extra</td>
<td>Shows the current value even if the value of p_value is not located in the select list.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to create a select list based on an LOV defined in the application.

```sql
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV_XL(2, job, 'JOB_FLOW_LOV') job FROM emp
```

19.16 SELECT_LIST_FROM_QUERY Function

This function dynamically generates a select list from a query. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```sql
APEX_ITEM.SELECT_LIST_FROM_QUERY(
    p_idx IN NUMBER,
    p_value IN VARCHAR2 DEFAULT NULL,
)
```
p_query IN VARCHAR2,
p_attributes IN VARCHAR2 DEFAULT NULL,
p_show_null IN VARCHAR2 DEFAULT 'YES',
p_null_value IN VARCHAR2 DEFAULT '%NULL%',
p_null_text IN VARCHAR2 DEFAULT '%',
p_item_id IN VARCHAR2 DEFAULT NULL,
p_item_label IN VARCHAR2 DEFAULT NULL,
p_show_extra IN VARCHAR2 DEFAULT 'YES')
RETURN VARCHAR2;

Parameters

Table 19-16 SELECT_LIST_FROM_QUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Current value. This value should be a value in the p_query parameter.</td>
</tr>
<tr>
<td>p_query</td>
<td>SQL query that is expected to select two columns, a display column, and a return column. For example:</td>
</tr>
<tr>
<td></td>
<td>SELECT dname, deptno FROM dept</td>
</tr>
<tr>
<td></td>
<td>Note that this is used only by the SELECT_LIST_FROM_QUERY function. Also note, if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_show_null</td>
<td>Extra select option to enable the NULL selection. Range of values is YES and NO.</td>
</tr>
<tr>
<td>p_null_value</td>
<td>Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;select&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
<tr>
<td>p_show_extra</td>
<td>Show the current value even if the value of p_value is not located in the select list.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates a select list based on a SQL query.

SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY(3, job, 'SELECT DISTINCT job FROM emp') job
FROM emp
19.17 SELECT_LIST_FROM_QUERY_XL Function

This function is the same as SELECT_LIST_FROM_QUERY, but its return value is a CLOB. This allows its use in SQL queries where you need to handle a column value longer than 4000 characters. Returned values will be limited to 32K. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT NULL,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT NULL,
    p_show_null     IN    VARCHAR2 DEFAULT 'YES',
    p_null_value    IN    VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN    VARCHAR2 DEFAULT '%',
    p_item_id       IN    VARCHAR2 DEFAULT NULL,
    p_item_label    IN    VARCHAR2 DEFAULT NULL,
    p_show_extra    IN    VARCHAR2 DEFAULT 'YES')
RETURN CLOB;

Parameters

Table 19-17  SELECT_LIST_FROM_QUERY_XL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Current value. This value should be a value in the p_query parameter.</td>
</tr>
<tr>
<td>p_query</td>
<td>SQL query that is expected to select two columns, a display column, and a return column. For example: SELECT dname, deptno FROM dept</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_show_null</td>
<td>Extra select parameters to enable the NULL selection. Range of values is YES and NO.</td>
</tr>
<tr>
<td>p_null_value</td>
<td>Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;select&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>
Table 19-17  (Cont.) SELECT_LIST_FROM_QUERY_XL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_show_extra</td>
<td>Show the current value even if the value of p_value is not located in the select list.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates a select list based on a SQL query.

```sql
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(3, job,'SELECT DISTINCT job FROM emp') job
FROM emp
```

19.18 SWITCH Function

This function dynamically generates flip toggle item. If On/Off value and label are not passed, it renders Yes/No toggle. Similar to other functions available in the APEX_ITEM package, switch function is designed to generate forms with F01 to F50 form array elements.

Syntax

```sql
APEX_ITEM.SWITCH(
    p_idx IN NUMBER,
    p_value IN VARCHAR2,
    p_on_value IN VARCHAR2 DEFAULT 'Y',
    p_on_label IN VARCHAR2 DEFAULT 'Yes',
    p_off_value IN VARCHAR2 DEFAULT 'N',
    p_off_label IN VARCHAR2 DEFAULT 'No',
    p_item_id IN VARCHAR2 DEFAULT NULL,
    p_item_label IN VARCHAR2,
    p_attributes IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 19-18  SWITCH Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.</td>
</tr>
<tr>
<td>p_value</td>
<td>Form element current value.</td>
</tr>
<tr>
<td>p_on_value</td>
<td>The value of the item if the user picks On option.</td>
</tr>
<tr>
<td>p_on_label</td>
<td>The display text for the On option.</td>
</tr>
<tr>
<td>p_off_value</td>
<td>The value of the item if the user picks Off option.</td>
</tr>
<tr>
<td>p_off_label</td>
<td>The display text for the Off option.</td>
</tr>
</tbody>
</table>
### Table 19-18  (Cont.) SWITCH Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;input&gt; tag. Try concatenating some string with rownum to make it unique.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional HTML attributes to use for the form item.</td>
</tr>
</tbody>
</table>

#### Example

The following example demonstrates the use of APEX_ITEM.SWITCH to generate a Yes/No flip toggle item where:

- A form array element F01 will be generated (p_idx parameter).
- The initial value for each element will be equal to N (p_value parameter).
- A HTML ID attribute will be generated for each row with the current rownum to uniquely identify. (p_item_id parameter). An ID of 'IS_MANAGER_2' is generated for row 2.)
- A HTML label element will be generated for each row (p_item_label parameter).

```sql
SELECT ename "Name",
APEX_ITEM.SWITCH (
    p_idx => 1,
    p_value => 'N',
    p_item_id => 'IS_MANAGER_'||rownum,
    p_item_label => apex_escape.html(ename)||': Is Manager' )
"Is Manager"
FROM emp;
```

### 19.19 TEXT Function

This function generates text fields (or text input form items) from a SQL query.

#### Syntax

```sql
APEX_ITEM.TEXT(
    p_idx IN NUMBER,
    p_value IN VARCHAR2 DEFAULT NULL,
    p_size IN NUMBER DEFAULT NULL,
    p_maxlength IN NUMBER DEFAULT NULL,
    p_attributes IN VARCHAR2 DEFAULT NULL,
    p_item_id IN VARCHAR2 DEFAULT NULL,
    p_item_label IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```
### Parameters

**Table 19-19  TEXT Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number to identify the item you want to generate. The number determines which <strong>G_FXX</strong> global is populated. See Also: &quot;APEX_APPLICATION&quot;</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of a text field item.</td>
</tr>
<tr>
<td>p_size</td>
<td>Controls HTML tag attributes (such as disabled).</td>
</tr>
<tr>
<td>p_maxlength</td>
<td>Maximum number of characters that can be entered in the text box.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the <code>&lt;input&gt;</code> tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

### 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.

```sql
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
```

### 19.20 TEXTAREA Function

This function creates text areas.

**Syntax**

```sql
APEX_ITEM.TEXTAREA(  
    p_idx IN NUMBER,  
    p_value IN VARCHAR2 DEFAULT NULL,  
    p_rows IN NUMBER DEFAULT 40,  
    p_cols IN NUMBER DEFAULT 4,  
    p_attributes IN VARCHAR2 DEFAULT NULL,
)
```
Parameters

Table 19-20  TEXTAREA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_idx</td>
<td>Number to identify the item you want to generate. The number determines which G_FXX global is populated.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;APEX_APPLICATION&quot;</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of the text area item.</td>
</tr>
<tr>
<td>p_rows</td>
<td>Height of the text area (HTML rows attribute)</td>
</tr>
<tr>
<td>p_cols</td>
<td>Width of the text area (HTML column attribute).</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Extra HTML parameters you want to add.</td>
</tr>
<tr>
<td>p_item_id</td>
<td>HTML attribute ID for the &lt;textarea&gt; tag.</td>
</tr>
<tr>
<td>p_item_label</td>
<td>Invisible label created for the item.</td>
</tr>
</tbody>
</table>

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
```

19.21 TEXT_FROM_LOV Function

Use this function to display an item as text, deriving the display value of the named LOV.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV (  
    p_value       IN    VARCHAR2 DEFAULT NULL,  
    p_lov         IN    VARCHAR2,  
    p_null_text   IN    VARCHAR2 DEFAULT '.defaultValue')  
RETURN VARCHAR2;
```

Parameters

Table 19-21  TEXT_FROM_LOV Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Value of a field item.</td>
</tr>
<tr>
<td></td>
<td>Note that if p_value is not located in the list of values, p_null_text is value displayed.</td>
</tr>
</tbody>
</table>
Table 19-21  (Cont.) TEXT_FROM_LOV Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lov</td>
<td>Text name of a shared list of values. This list of values must be defined in your application.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>Value displayed when the value of the field item is NULL.</td>
</tr>
</tbody>
</table>

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
```

19.22 TEXT_FROM_LOV_QUERY Function

Use this function to display an item as text, deriving the display value from a list of values query.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV_QUERY (p_value IN VARCHAR2 DEFAULT NULL,
                                 p_query  IN VARCHAR2,
                                 p_null_text IN VARCHAR2 DEFAULT '§')
RETURN VARCHAR2;
```

Parameters

Table 19-22  TEXT_FROM_LOV_QUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Value of a field item.</td>
</tr>
<tr>
<td>p_query</td>
<td>SQL query that is expected to select two columns, a display column and a return column. For example:</td>
</tr>
<tr>
<td></td>
<td>SELECT dname, deptno FROM dept</td>
</tr>
<tr>
<td></td>
<td>Note if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.</td>
</tr>
<tr>
<td>p_null_text</td>
<td>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.</td>
</tr>
</tbody>
</table>
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
```
The APEX_JAVASCRIPT package provides utility functions for adding dynamic JavaScript code to HTTP output. This package is usually used for plug-in development.

- ADD_3RD_PARTY_LIBRARY_FILE Procedure
- ADD_ATTRIBUTE Function Signature 1
- ADD_ATTRIBUTE Function Signature 2
- ADD_ATTRIBUTE Function Signature 3
- ADD_ATTRIBUTE Function Signature 4
- ADD_INLINE_CODE Procedure
- ADD_LIBRARY Procedure
- ADD_REQUIREJS Procedure
- ADD_REQUIREJS_DEFINE Procedure
- ADD_ONLOAD_CODE Procedure
- ADD_VALUE Function Signature 1
- ADD_VALUE Function Signature 2
- ADD_VALUE Function Signature 3
- ADD_VALUE Function Signature 4
- Escape Function

### 20.1 ADD_3RD_PARTY_LIBRARY_FILE Procedure

This procedure adds the script tag to load a 3rd party javascript library file and also takes into account the specified Content Delivery Network for the application. Supported libraries include: jQuery, jQueryUI, and jQuery Mobile.

**Syntax**

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

Table 20-1  ADD_3RD_PARTY_LIBRARY_FILE Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_library</td>
<td>Use one of the c_library_* constants</td>
</tr>
<tr>
<td>p_file_name</td>
<td>Specifies the file name without version, .min and .js</td>
</tr>
<tr>
<td>p_directory</td>
<td>Directory where the file p_file_name is located (optional)</td>
</tr>
<tr>
<td>p_version</td>
<td>If no value is provided then the same version Application Express ships is used (optional)</td>
</tr>
</tbody>
</table>

Example

This example loads the JavaScript file of the Draggable feature of jQuery UI.

```javascript
apex_javascript.add_3rd_party_library_file (
    p_library   =>apex_javascript.c_library_jquery_ui,
    p_file_name => 'jquery.ui.draggable')
```

20.2 ADD_ATTRIBUTE Function Signature 1

This function returns the attribute and the attribute's escaped text surrounded by double quotation marks.

**Note:**

This function does not escape HTML tags. It only prevents HTML tags from breaking the JavaScript object attribute assignment. To prevent XSS (cross site scripting) attacks, you must also call `SYS.HTF.ESCAPE_SC` to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```sql
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name       IN VARCHAR2,
    p_value      IN VARCHAR2,
    p_omit_null  IN BOOLEAN:=TRUE,
    p_add_comma  IN BOOLEAN:=TRUE)
RETURN VARCHAR2;
```
Parameters

Table 20-2  ADD_ATTRIBUTE Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_value</td>
<td>Text to be assigned to the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_omit_null</td>
<td>If set to TRUE and p_value is empty, returns NULL.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If set to TRUE, a trailing comma is added when a value is returned.</td>
</tr>
</tbody>
</table>

Example

Adds a call to the addEmployee JavaScript function and passes in a JavaScript object with different attribute values. The output of this call looks like:

```javascript
addEmployee(
    {"FirstName":"John",
     "LastName":"Doe",
     "Salary":2531.29,
     "Birthday":new Date(1970,1,15,0,0,0),
     "isSalesman":true
    });
```

As the last attribute you should use the parameter combination FALSE (p_omit_null), FALSE (p_add_comma) so that the last attribute is always generated. This avoids that you have to check for the other parameters if a trailing comma should be added or not.

```javascript
apex_javascript.add_onload_code (  
    'addEmployee(',  
     '{',  
     apex_javascript.add_attribute('FirstName',  
        sys.htf.escape_sc(l_first_name))  
    ||  
    apex_javascript.add_attribute('LastName',  
        sys.htf.escape_sc(l_last_name))  
    ||  
    apex_javascript.add_attribute('Salary',  
        l_salary)  
    ||  
    apex_javascript.add_attribute('Birthday',  
        l_birthday)  
    ||  
    apex_javascript.add_attribute('isSalesman',  
        l_is_salesman, false,  
        false)  
    ',  
    '}');
```

20.3 ADD_ATTRIBUTE Function Signature 2

This function returns the attribute and the attribute's number.

Syntax

```sql
APEX_JAVASCRIPT.ADD_ATTRIBUTE (  
    p_name       IN VARCHAR2,  
    p_value      IN NUMBER,
```
Parameters

Table 20-3  ADD_ATTRIBUTE Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_value</td>
<td>Number which should be assigned to the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_omit_null</td>
<td>If set to TRUE and p_value is empty, returns NULL.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If set to TRUE, a trailing comma is added when a value is returned.</td>
</tr>
</tbody>
</table>

Example

See example for ADD_ATTRIBUTE Function Signature 1.

20.4 ADD_ATTRIBUTE Function Signature 3

This function returns the attribute and a JavaScript boolean of TRUE, FALSE, or NULL.

Syntax

APEX_JAVASCRIPT.ADD_ATTRIBUTE {
    p_name       IN VARCHAR2,
    p_value      IN BOOLEAN,
    p_omit_null  IN BOOLEAN:=TRUE,
    p_add_comma  IN BOOLEAN:=TRUE)
RETURN VARCHAR2;

Parameters

Table 20-4  ADD_ATTRIBUTE Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_value</td>
<td>Boolean assigned to the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_omit_null</td>
<td>If p_omit_null is TRUE and p_value is NULL the function returns NULL.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If set to TRUE a trailing comma is added when a value is returned.</td>
</tr>
</tbody>
</table>

Example

See example for ADD_ATTRIBUTE Function Signature 1
20.5 ADD_ATTRIBUTE Function Signature 4

This function returns the attribute and the attribute's date. If p_value is null the value null is returned.

Syntax

APEX_JAVASCRIPT.ADD_ATTRIBUTE (  
  p_name       IN VARCHAR2,  
  p_value      IN DATE,  
  p_omit_null  IN BOOLEAN:=TRUE,  
  p_add_comma  IN BOOLEAN:=TRUE)  
RETURN VARCHAR2;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_value</td>
<td>Date assigned to the JavaScript object attribute.</td>
</tr>
<tr>
<td>p_omit_null</td>
<td>If p_omit_null is TRUE and p_value is NULL the function returns NULL.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If set to TRUE a trailing comma is added when a value is returned.</td>
</tr>
</tbody>
</table>

Example

See example for ADD_ATTRIBUTE Function Signature 1

20.6 ADD_INLINE_CODE Procedure

This procedure adds a code snippet that is included inline into the HTML output. For example, you can use this procedure to add new functions or global variable declarations.

Note:

If you want to execute code you should use ADD_ONLOAD_CODE Procedure.

Syntax

APEX_JAVASCRIPT.ADD_INLINE_CODE (  
  p_code       IN VARCHAR2,  
  p_key        IN VARCHAR2 DEFAULT NULL);
Parameters

Table 20-6  ADD_INLINE_CODE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_code</td>
<td>JavaScript code snippet. For example: $s('P1_TEST',123);</td>
</tr>
<tr>
<td>p_key</td>
<td>Identifier for the code snippet. If specified and a code snippet with the same name has already been added, the new code snippet is ignored. If p_key is NULL the snippet is always added.</td>
</tr>
</tbody>
</table>

Example

The following example includes the JavaScript function initMySuperWidget in the HTML output. If the plug-in is used multiple times on the page and the add_inline_code is called multiple times, it is added once to the HTML output because all calls have the same value for p_key.

```javascript
apex_javascript.add_inline_code(
  p_code => 'function initMySuperWidget(){'||chr(10)||
    ' // do something'||chr(10)||
    '}',
  p_key => 'my_super_widget_function');
```

20.7 ADD_LIBRARY Procedure

This procedure adds the script tag to load a JavaScript library. If a library has been added, it is not added a second time.

Syntax

```sql
APEX_JAVASCRIPT.ADD_LIBRARY(
  p_name                    IN VARCHAR2,
  p_directory               IN VARCHAR2,
  p_version                 IN VARCHAR2 DEFAULT NULL,
  p_check_to_add_minified   IN BOOLEAN DEFAULT FALSE,
  p_skip_extension          IN BOOLEAN  DEFAULT FALSE,
  p_ie_condition            IN VARCHAR2 DEFAULT NULL,
  p_requirejs_module        IN VARCHAR2 DEFAULT NULL,
  p_requirejs_js_expression IN VARCHAR2 DEFAULT NULL,
  p_requirejs_required      IN BOOLEAN DEFAULT FALSE,
  p_key                     IN VARCHAR2 DEFAULT NULL);
```
Parameters

Table 20-7 ADD_LIBRARY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the JavaScript file. Must not use .js when specifying.</td>
</tr>
<tr>
<td>p_directory</td>
<td>Directory where JavaScript library is loaded. Must have a trailing slash.</td>
</tr>
<tr>
<td>p_version</td>
<td>Version identifier.</td>
</tr>
<tr>
<td>p_check_to_add_minified</td>
<td>If TRUE, the procedure tests if it is appropriate to add .min extension and add it if appropriate. This is added if an application is not running in DEBUG mode, and omitted when in DEBUG mode.</td>
</tr>
<tr>
<td>p_skip_extension</td>
<td>If TRUE the extension .js is NOT added.</td>
</tr>
<tr>
<td>p_ie_condition</td>
<td>Condition which is used as Internet Explorer condition.</td>
</tr>
<tr>
<td>p_requirejs_module</td>
<td>Module name which is used to expose the library to RequireJS.</td>
</tr>
<tr>
<td>p_requirejs Js_expression</td>
<td>JavaScript expression which is used to expose the library to the RequireJS module.</td>
</tr>
<tr>
<td>p_requirejs_required</td>
<td>This has to be true if the library uses RequireJS in its code to loading other JavaScript files.</td>
</tr>
<tr>
<td>p_key</td>
<td>Name used to indicate if the library has already been loaded. If not specified, defaults to p_directory</td>
</tr>
</tbody>
</table>

Example

The following example includes the JavaScript library file named hammer-2.0.4.min.js (if the application has not been started from the Builder), or hammer-2.0.4.js (if the application has been started from the Builder or is running in DEBUG mode), from the directory specified by p_plugin.file_prefix. Since p_skip_extension is not specified, this defaults to .js. Also, since p_key is not specified, the key defaults to p_plugin.file_prefix||hammer-2.0.4. Hammer is a JavaScript library which exposes itself to RequireJS using hammerjs as module name.

```javascript
apex_javascript.add_library (  
  p_name => 'hammer-2.0.4#MIN#',  
  p_directory => p_plugin.file_prefix,  
  p_requirejs_module => 'hammerjs',  
  p_requirejs_js_expression => 'Hammer' );
```

20.8 ADD_REQUIREJS Procedure

This procedure adds the script tag to load the RequireJS library.
Syntax

procedure add_requirejs;

20.9 ADD_REQUIREJS_DEFINE Procedure

This procedure adds a RequireJS define after RequireJS has been loaded to let it know about the existence of a library.

Syntax

APEX_JAVASCRIPT.add_requirejs_define (  
    p_module in varchar2,  
    p_js_expression in varchar2 );

Parameters

Table 20-8    ADD_REQUIREJS_DEFINE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_module</td>
<td></td>
</tr>
<tr>
<td>p_js_expression</td>
<td></td>
</tr>
</tbody>
</table>

Example

apex_javascript.add_requirejs_define (  
    p_module => 'hammerjs',  
    p_js_expression => 'Hammer' );

20.10 ADD_ONLOAD_CODE Procedure

This procedure adds a javascript code snippet to the HTML output which is executed by the onload event. If an entry with the same key exists it is ignored. If p_key is NULL the snippet is always added.

Syntax

APEX_JAVASCRIPT.ADD_ONLOAD_CODE (  
    p_code IN VARCHAR2,  
    p_key IN VARCHAR2 DEFAULT NULL);

Parameters

Table 20-9    ADD_ONLOAD_CODE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_code</td>
<td>Javascript code snippet to be executed during the onload event.</td>
</tr>
</tbody>
</table>
Table 20-9  (Cont.) ADD_ONLOAD_CODE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_key</td>
<td>Any name to identify the specified code snippet. If specified, the code snippet is added if there has been no other call with the same p_key. If p_key is NULL the code snippet is always added.</td>
</tr>
</tbody>
</table>

Example

Adds the JavaScript call `initMySuperWidget()` to the onload buffer. If the plug-in is used multiple times on the page and the add_onload_code is called multiple times, it is added once to the HTML output because all calls have the same value for p_key.

```java
apexJavascript.add_onload_code (p_code => 'initMySuperWidget();'
p_key  => 'my_super_widget');
```

20.11 ADD_VALUE Function Signature 1

This function returns the escaped text surrounded by double quotation marks. For example, this string could be returned "That's a test".

Note:

This function does not escape HTML tags. It only prevents HTML tags from breaking the JavaScript object attribute assignment. To prevent XSS (cross site scripting) attacks, you must also call SYS.HTF.ESCAPE_SC to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```java
APEX_JAVASCRIPT.ADD_VALUE (p_value          IN VARCHAR2,
p_add_comma      IN BOOLEAN :=TRUE)
RETURN VARCHAR2;
```

Parameters

Table 20-10  ADD_VALUE Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Text to be escaped and wrapped by double quotation marks.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If p_add_comma is TRUE a trailing comma is added.</td>
</tr>
</tbody>
</table>
Example

This example adds some JavaScript code to the onload buffer. The value of
`p_item.attribute_01` is first escaped with `htf.escape_sc` to prevent XSS attacks and
then assigned to the JavaScript variable `lTest` by calling
`apex_javascript.add_value`. `Add_value` takes care of properly escaping the value
and wrapping it with double quotation marks. Because commas are not wanted,
`p_add_comma` is set to FALSE.

```javascript
apex_javascript.add_onload_code {
    'var lTest = '||
    apex_javascript.add_value(sys.htf.escape_sc(p_item.attribute_01),
    FALSE)||';'||chr(10)||
    'showMessage(lTest);' );
}
```

20.12 ADD_VALUE Function Signature 2

This function returns `p_value` as JavaScript number, if `p_value` is NULL the value null
is returned.

Syntax

```sql
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN NUMBER,
    p_add_comma      IN BOOLEAN :=TRUE)
RETURN VARCHAR2;
```

Parameters

**Table 20-11   ADD_VALUE Signature 2 Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Number which should be returned as JavaScript number.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If <code>p_add_comma</code> is TRUE a trailing comma is added. Default is TRUE.</td>
</tr>
</tbody>
</table>

Example

See example for ADD_VALUE Function Signature 1.

20.13 ADD_VALUE Function Signature 3

This function returns `p_value` as JavaScript boolean. If `p_value` is NULL the value null
is returned.

Syntax

```sql
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN BOOLEAN,
```
Parameters

Table 20-12  ADD_VALUE Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Boolean which should be returned as JavaScript boolean.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If p_add_comma is TRUE a trailing comma is added. Default is TRUE.</td>
</tr>
</tbody>
</table>

Example

See example for ADD_VALUE Function Signature 1 .

### 20.14 ADD_VALUE Function Signature 4

This function returns p_value as JavaScript date object, if p_value is NULL the value null is returned.

Syntax

APEX_JAVASCRIPT.ADD_VALUE (  
    p_value          IN NUMBER,  
    p_add_comma      IN BOOLEAN :=TRUE)  
RETURN VARCHAR2;

Parameters

Table 20-13  ADD_VALUE Signature 4 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Date which should be returned as JavaScript date object.</td>
</tr>
<tr>
<td>p_add_comma</td>
<td>If p_add_comma is TRUE a trailing comma is added. Default is TRUE.</td>
</tr>
</tbody>
</table>

Example

See example for ADD_VALUE Function Signature 1 .

### 20.15 Escape Function

This function escapes text to be used in JavaScript. This function uses APEX_ESCAPE.JS_LITERAL to escape characters and provide a reference to that other API.
Note:

This function prevents HTML tags from breaking the JavaScript object attribute assignment and also escapes the HTML tags '<' and '>'. It does not escape other HTML tags, therefore to be sure to prevent XSS (cross site scripting) attacks, you must also call SYS.HTF.ESCAPE_SC to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

APEX_JAVASCRIPT.ESCAPE (  
  p_text  IN VARCHAR2)  
RETURN VARCHAR2;

Parameters

Table 20-14  ESCAPE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_text</td>
<td>Text to be escaped.</td>
</tr>
</tbody>
</table>

Example

Adds some JavaScript code to the onload buffer. The value of p_item.attribute_01 is first escaped with htf.escape_sc to prevent XSS attacks and then escaped with apex_javascript.escape to prevent that special characters like a quotation mark break the JavaScript code.

```javascript
apexJavascript.add_onload_code (  
  'var lTest = ""'||
apexJavascript.escape(sys.htf.escape_sc(p_item.attribute_01))||"";'||
chr(10)||
  'showMessage(lTest);' );
```
This package includes utilities that parse and generate JSON.

- Package Overview and Examples
- Constants and Data Types
- CLOSE_ALL Procedure
- CLOSE_ARRAY Procedure
- CLOSE_OBJECT Procedure
- DOES_EXIST Function
- FIND_PATHS_LIKE Function
- FREE_OUTPUT Procedure
- FLUSH Procedure
- GET_BOOLEAN Function
- GET_CLOB_OUTPUT Function
- GET_COUNT Function
- GET_DATE Function
- GET_MEMBERS Function
- GET_NUMBER Function
- GET_VALUE Function
- GET_T_NUMBER Function
- GET_T_VARCHAR2 Function
- GET_VARCHAR2 Function
- GET_CLOB Function
- INITIALIZE_CLOB_OUTPUT Procedure
- INITIALIZE_OUTPUT Procedure
- OPEN_ARRAY Procedure
- OPEN_OBJECT Procedure
- PARSE Procedure Signature 1
- PARSE Procedure Signature 2
- STRINGIFY Function Signature 1
- STRINGIFY Function Signature 2
- STRINGIFY Function Signature 3
- STRINGIFY Function Signature 4
- TO_MEMBER_NAME Function
To read from a string that contains JSON data, first use `parse()` to convert the string to an internal format. Then use the `get_` routines (e.g. `get_varchar2()`, `get_number()`, ...) to access the data and `find_paths_like()` to search. Alternatively, use `to_xmltype()` to convert a JSON string to an xmltype.

This package also contains procedures to generate JSON-formatted output. Use the overloaded `open_`(), `close_`() and `write()` procedures for writing.

**Example 1**

This example parses a JSON string and prints the value of member variable "a".

```sql
DECLARE
    s varchar2(32767) := '{ "a": 1, "b": ["hello", "world"]};
BEGIN
    apex_json.parse(s);
    sys.dbms_output.put_line('a' is '|| apex_json.get_varchar2(p_path => 'a'));
END;
```
Chapter 21

Constants and Data Types

Example 2
This example converts a JSON string to XML and uses XMLTABLE to query member
values.
select col1, col2
from xmltable (
'/json/row'
passing apex_json.to_xmltype('[{"col1": 1, "col2": "hello"},'||
'{"col1": 2, "col2": "world"}]')
columns
col1 number path '/row/col1',
col2 varchar2(5) path '/row/col2' );
Example 3
This example writes a nested JSON object to the HTP buffer.
BEGIN
apex_json.open_object;
-- {
apex_json.write('a', 1);
-- "a":1
apex_json.open_array('b'); -- ,"b":[
apex_json.open_object;
-{
apex_json.write('c',2); -"c":2
apex_json.close_object; -}
apex_json.write('hello'); -- ,"hello"
apex_json.write('world'); -- ,"world"
apex_json.close_all;
-- ]
-- }
END;

21.2 Constants and Data Types
Parser Interface
The following are constants used for the parser interface:
subtype t_kind is binary_integer range 1 .. 8;
c_null
constant t_kind := 1;
c_true
constant t_kind := 2;
c_false
constant t_kind := 3;
c_number constant t_kind := 4;
c_varchar2 constant t_kind := 5;
c_object constant t_kind := 6;
c_array
constant t_kind := 7;
c_clob
constant t_kind := 8;
Storage for JSON Data
JSON data is stored in an index by varchar2 table. The JSON values are stored as
records. The discriminator "kind" determines whether the value is null, true, false, a
number, a varchar2, a clob, an object or an array. It depends on "kind" which record

21-3


fields are used and how. If not explicitly mentioned below, the other record fields' values are undefined:

* c_null: -
* c_true: -
* c_false: -
* c_number: number_value contains the number value
* c_varchar2: varchar2_value contains the varchar2 value
* c_clob: clob_value contains the clob
* c_object: object_members contains the names of the object's members
* c_array: number_value contains the array length

type t_value is record (  
    kind           t_kind,
    number_value   number,
    varchar2_value varchar2(32767),
    clob_value clob,
    object_members apex_t_varchar2 );
type t_values is table of t_value index by varchar2(32767);

Default Format for Dates

c_date_iso8601 constant varchar2(30) := 'yyyy-mm-dd"T"hh24:mi:ss"Z"';

Default JSON Values Table

g_values t_values;

Errors Thrown for PARSE()

e_parse_error     exception;
pragma exception_init(e_parse_error, -20987);

21.3 CLOSE_ALL Procedure

This procedure closes all objects and arrays up to the outermost nesting level.

Syntax

APEX_JSON.CLOSE_ALL;

Parameters

None.
Example
See "Package Overview and Examples".

21.4 CLOSE_ARRAY Procedure
This procedure writes a close bracket symbol as follows:
]

Syntax
APEX_JSON.CLOSE_ARRAY ();

Parameters
None.

Example
See "Package Overview and Examples".

21.5 CLOSE_OBJECT Procedure
This procedure writes a close curly bracket symbol as follows:
}

Syntax
APEX_JSON.CLOSE_OBJECT ();

Parameters
None.

Example
See "Package Overview and Examples".

21.6 DOES_EXIST Function
This function determines whether the given path points to an existing value.

Syntax
APEX_JSON.DOES_EXIST ( 
  p_path             IN VARCHAR2,
  p0                 IN VARCHAR2 DEFAULT NULL,
  p1                 IN VARCHAR2 DEFAULT NULL,
  p2                 IN VARCHAR2 DEFAULT NULL, 
);
p3 IN VARCHAR2 DEFAULT NULL,
p4 IN VARCHAR2 DEFAULT NULL,
p_values IN t_values DEFAULT g_values )
RETURN BOOLEAN;

Parameters

Table 21-1  DOES_EXIST Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns

Table 21-2  DOES_EXIST Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Given path points to an existing value.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Given path does not point to an existing value</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints whether it contains values under a path.

```sql
DECLARE
  j apex_json.t_values;
BEGIN
  apex_json.parse(j, '{ "items": [ 1, 2, { "foo": true } ] }');
  if apex_json.does_exist(p_path => 'items[%d].foo', p0 => 3, p_values => j) then
    dbms_output.put_line('found items[3].foo');
  end if;
END;
```

21.7 FIND_PATHS_LIKE Function

This function returns paths into p_values that match a given pattern.

Syntax

```sql
APEX_JSON.FIND_PATHS_LIKE ( 
  p_return_path IN VARCHAR2,
p_subpath IN VARCHAR2 DEFAULT NULL,
p_value IN VARCHAR2 DEFAULT NULL,
)"
p_values IN t_values DEFAULT g_values )
RETURN apex_t_varchar2;

Parameters

Table 21-3  FIND_PATHS_LIKE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_return_path</td>
<td>Search pattern for the return path.</td>
</tr>
<tr>
<td>p_subpath</td>
<td>Search pattern under p_return_path (optional).</td>
</tr>
<tr>
<td>p_value</td>
<td>Search pattern for value (optional).</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns/Raised Errors

Table 21-4  FIND_PATHS_LIKE Function Returns and Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apex_t_varchar2</td>
<td>Table of paths that match the pattern.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values (p_path) is not an array or object.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string, finds paths that match a pattern, and prints the values under the paths.

DECLARE
  j     apex_json.t_values;
  l_paths apex_t_varchar2;
BEGIN
  apex_json.parse(j, '{ "items": [ { "name": "Amulet of Yendor", "magical": true }, ...
                          { "name": "Slippers", "magical": "rather not" } ] }');
  l_paths := apex_json.find_paths_like (p_values => j, p_return_path => 'items[%]', p_subpath => '.magical', p_value => 'true');
  dbms_output.put_line('Magical items:');
  for i in 1 .. l_paths.count loop
    dbms_output.put_line(apex_json.get_varchar2(p_values => j, p_path => l_paths(i)||'.name'));
  end loop;
END;
21.8 FREE_OUTPUT Procedure

Frees output resources. Call this procedure after process if you are using INITIALIZE_CLOB_OUTPUT to write to a temporary CLOB.

Syntax

free_output;

Example

This example configures APEX_JSON for CLOB output, generate JSON, print the CLOB with DBMS_OUTPUT, and finally free the CLOB.

BEGIN
    apex_json.initialize_clob_output;

    apex_json.open_object;
    apex_json.write('hello', 'world');
    apex_json.close_object;

    dbms_output.put_line(apex_json.get_clob_output);

    apex_json.free_output;
END;

21.9 FLUSH Procedure

This procedure flushes pending changes. Note that close procedures automatically flush.

Syntax

APEX_JSON.FLUSH

Parameters

None.

Example

This example writes incomplete JSON.

BEGIN
    apex_json.open_object;
    apex_json.write('attr', 'value');
    apex_json.flush;
    sys.htp.p('the "" is missing');
END;
21.10 GET_BOOLEAN Function

This function returns a boolean number value.

Syntax

APEX_JSON.GET_BOOLEAN (  
  p_path             IN VARCHAR2,  
  p0                 IN VARCHAR2 DEFAULT NULL,  
  p1                 IN VARCHAR2 DEFAULT NULL,  
  p2                 IN VARCHAR2 DEFAULT NULL,  
  p3                 IN VARCHAR2 DEFAULT NULL,  
  p4                 IN VARCHAR2 DEFAULT NULL,  
  p_default          IN BOOLEAN  DEFAULT NULL,  
  p_values           IN t_values DEFAULT g_values )  
RETURN BOOLEAN;

Parameters

Table 21-5  GET_BOOLEAN Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by</td>
</tr>
<tr>
<td></td>
<td>the p[i-1].</td>
</tr>
<tr>
<td>p_default</td>
<td>The default value if the member does not exist.</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns

Table 21-6  GET_BOOLEAN Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Value at the given path position.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Value at the given path position.</td>
</tr>
<tr>
<td>NULL</td>
<td>Value at the given path position.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values (p_path) is not boolean.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints the boolean value at a position.

DECLARE  
  j apex_json.t_values;  
BEGIN  
apex_json.parse(j, '{ "items": [ 1, 2, { "foo": true } ] }');  
  if apex_json.get_boolean(p_path=>'items[%d].foo', p0=>3, p_values=>j)
then
    dbms_output.put_line('items[3].foo is true');
END IF;
END;

21.11 GET_CLOB_OUTPUT Function

Returns the temporary CLOB that you created with INITIALIZE_CLOB_OUTPUT.

Syntax

function get_clob_output
    return clob;

Example

This example configures APEX_JSON for CLOB output, generate JSON, print the CLOB with DBMS_OUTPUT, and finally free the CLOB.

BEGIN
    apex_json.initialize_clob_output;
    apex_json.open_object;
    apex_json.write('hello', 'world');
    apex_json.close_object;
    dbms_output.put_line(apex_json.get_clob_output);
    apex_json.free_output;
END;

21.12 GET_COUNT Function

This function returns the number of array elements or object members.

Syntax

APEX_JSON.GET_COUNT {
    p_path            IN VARCHAR2,
    p0                IN VARCHAR2 DEFAULT NULL,
    p1                IN VARCHAR2 DEFAULT NULL,
    p2                IN VARCHAR2 DEFAULT NULL,
    p3                IN VARCHAR2 DEFAULT NULL,
    p4                IN VARCHAR2 DEFAULT NULL,
    p_values          IN t_values DEFAULT g_values )
RETURN NUMBER;
Parameters

Table 21-7  GET_COUNT Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns/Raised Errors

Table 21-8  GET_COUNT Function Returns and Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>The number of array elements or object members or null if the array or object could not be found</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values (p_path) is not an array or object.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints the number of members at positions.

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, ©{ "foo": 3, "bar": [1, 2, 3, 4] ©});
    dbms_output.put_line(apex_json.get_count(p_path=>®.®.®p_values=>j)); -- 2 (foo and bar)
    dbms_output.put_line(apex_json.get_count(p_path=>®bar®.®p_values=>j));
    -- 4
END;

21.13 GET_DATE Function

This function returns a date member value.

Syntax

APEX_JSON.GET_DATE ( p_path IN VARCHAR2, p0 IN VARCHAR2 DEFAULT NULL, p1 IN VARCHAR2 DEFAULT NULL, p2 IN VARCHAR2 DEFAULT NULL, p3 IN VARCHAR2 DEFAULT NULL, p4 IN VARCHAR2 DEFAULT NULL, p_default IN DATE DEFAULT NULL, p_format IN VARCHAR2 DEFAULT c_date_iso8601, )
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_default</td>
<td>The default value if the member does not exist.</td>
</tr>
<tr>
<td>p_format</td>
<td>The date format mask.</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns/Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>.Returns the date.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values(p_path) is not a date.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints the value at a position.

```sql
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, ©{ "items": [ 1, 2, { "foo": "2014-04-29T10:08:00Z" }] });
    dbms_output.put_line(to_char(apex_json.get_date(p_path=>©items[%d].foo©,p0=>3, p_values=>j), ©DD-Mon-YYYY©));
END;
```

21.14 GET_MEMBERS Function

This function returns the table of OBJECT_MEMBERS names for an object.

Syntax

```sql
APEX_JSON.GET_MEMBERS ( p_path IN VARCHAR2,
                        p0    IN VARCHAR2 DEFAULT NULL,
                        p1    IN VARCHAR2 DEFAULT NULL,
                        p2    IN VARCHAR2 DEFAULT NULL,
                        ...
) RETURN OBJECT_MEMBERS;
```
p3 IN VARCHAR2 DEFAULT NULL,
p4 IN VARCHAR2 DEFAULT NULL,
p_values IN t_values DEFAULT g_values ) RETURN APEX_T_VARCHAR2;

Parameters

Table 21-11 GET_MEMBERS Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns/Raised Errors

Table 21-12 GET_MEMBERS Function Returns and Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_MEMBERS</td>
<td>The OBJECT_MEMBERS of the object or null if the object could not be found.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values(p_path) is not an array or object.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints members at positions.

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, ©{ "foo": 3, "bar": [1, 2, 3, 4] ©});
    dbms_output.put_line(apex_json.get_members(p_path=>'.',p_values=>j)(1)); -- foo
    dbms_output.put_line(apex_json.get_members(p_path=>'.',p_values=>j)(2)); -- bar
END;

21.15 GET_NUMBER Function

This function returns a numeric number value.

Syntax

APEX_JSON.GET_NUMBER ( 
p_path IN VARCHAR2,
p0 IN VARCHAR2 DEFAULT NULL,
p1 IN VARCHAR2 DEFAULT NULL,
p2 IN VARCHAR2 DEFAULT NULL,
p3                 IN VARCHAR2 DEFAULT NULL,
p4                 IN VARCHAR2 DEFAULT NULL,
p_default          IN BOOLEAN DEFAULT NULL,
p_values           IN t_values DEFAULT g_values )
RETURN NUMBER;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_default</td>
<td>The default value if the member does not exist.</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns/Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>The value at the given path position.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values(p_path) is not a number.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints the value at a position.

```sql
DECLARE
  j apex_json.t_values;
BEGIN
  apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
  dbms_output.put_line(apex_json.get_number(p_path=>'items[%d].foo',p0=>3,p_values=>j));
END;
```

21.16 GET_VALUE Function

This function returns the t_value.

Syntax

```
APEX_JSON.GET_VALUE ( 
  p_path             IN VARCHAR2,
p0                  IN VARCHAR2 DEFAULT NULL,
p1                  IN VARCHAR2 DEFAULT NULL,
p2                  IN VARCHAR2 DEFAULT NULL,
p3                  IN VARCHAR2 DEFAULT NULL,
```
p4  IN VARCHAR2 DEFAULT NULL,
p_values  IN t_values DEFAULT g_values )
RETURN t_value;

Parameters

Table 21-15  GET_VALUE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by p[i-1].</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns/Raised Errors

Table 21-16  GET_VALUE Function Returns and Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_value</td>
<td>The t_value at the given path position. The record attributes are null if no data is found.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values (p_path) is not an array or object.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints attributes of values at positions.

```sql
DECLARE
  j apex_json.t_values;
  v apex_json.t_value;
BEGIN
  apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
  v := apex_json.get_value(p_path=>'bar[%d]',p0=> 2,p_values=>j); -- returns the t_value for bar[2]
  dbms_output.put_line(v.number_value); -- 2
  v := apex_json.get_value(p_path=>'does.not.exist',p_values=>j);
  dbms_output.put_line(case when v.kind is null then 'not found!' end);
END;
```

21.17 GET_T_NUMBER Function

This function returns the numeric attributes of an array.

Syntax

```sql
function get_t_number ( 
    p_path     in varchar2,
    p0         in varchar2 default null,
    p1         in varchar2 default null,
```
Parameters

Table 21-17  GET_T_NUMBER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is p_values.</td>
</tr>
</tbody>
</table>

Returns

Array member values if the referenced t_value is an array. An array with just the referenced value if it's type can be converted to a number.

Table 21-18  GET_T_NUMBER Function Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE_ERROR</td>
<td>On conversion errors.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and prints the value at position 1.

```sql
declare
    j   apex_json.t_values;
    l_elements apex_t_number;
begin
    apex_json.parse(j, '{ "foo": [111, 222], "bar": 333 }');
    l_elements := apex_json.get_t_number (
        p_values => j,
        p_path   => 'foo');
    for i in 1 .. l_elements.count loop
        sys.dbms_output.put_line(i||':'||l_elements(i));
    end loop;
    l_elements := apex_json.get_t_number (
        p_values => j,
        p_path   => 'bar');
    for i in 1 .. l_elements.count loop
        sys.dbms_output.put_line(i||':'||l_elements(i));
    end loop;
end;
```

Output:

```
1:111
```
21.18 GET_T_VARCHAR2 Function

This function returns the varchar2 attributes of an array.

Syntax

function get_t_varchar2 ( 
   p_path            in varchar2,
   p0                in varchar2 default null,
   p1                in varchar2 default null,
   p2                in varchar2 default null,
   p3                in varchar2 default null,
   p4                in varchar2 default null,
   p_values          in t_values default g_values ) 
return wwv_flow_t_varchar2;

Parameters

Table 21-19 GET_T_VARCHAR2 Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>

Returns

Array member values if the referenced t_value is an array. An array with just the referenced value if it's type can be converted to a varchar2.

Raises

Table 21-20 GET_T_VARCHAR2 Function Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE_ERROR</td>
<td>On conversion errors.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON and prints the value at position 1.

```java
declare
   j       apex_json.t_values;
   l_elements apex_t_varchar2;
begin
   apex_json.parse(j, '{"foo": ["one", "two"], "bar": "three" }');
```

ORACLE
l_elements := apex_json.get_t_varchar2 (  
    p_values => j,  
    p_path   => 'foo' );
for i in 1 .. l_elements.count loop  
    sys.dbms_output.put_line(i||'::'||l_elements(i));
end loop;

l_elements := apex_json.get_t_varchar2 (  
    p_values => j,  
    p_path   => 'bar' );
for i in 1 .. l_elements.count loop  
    sys.dbms_output.put_line(i||'::'||l_elements(i));
end loop;
end;

Output:  
1:one  
2:two  
1:three

21.19 GET_VARCHAR2 Function

This function returns a varchar2 member value. This function converts boolean and number values to varchar2 values.

Syntax

```apl
APEX_JSON.GET_VARCHAR2 (  
    p_path             IN VARCHAR2,  
    p0                 IN VARCHAR2 DEFAULT NULL,  
    p1                 IN VARCHAR2 DEFAULT NULL,  
    p2                 IN VARCHAR2 DEFAULT NULL,  
    p3                 IN VARCHAR2 DEFAULT NULL,  
    p4                 IN VARCHAR2 DEFAULT NULL,  
    p_default          IN BOOLEAN  DEFAULT NULL,  
    p_values           IN t_values DEFAULT g_values )  
RETURN VARCHAR2;
```

Parameters

Table 21-21  GET_VARCHAR2 Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by</td>
</tr>
<tr>
<td></td>
<td>the p[i-1].</td>
</tr>
<tr>
<td>p_default</td>
<td>The default value if the member does not exist.</td>
</tr>
<tr>
<td>p_values</td>
<td>Parsed JSON members. The default is g_values.</td>
</tr>
</tbody>
</table>
Returns/Raised Errors

Table 21-22  GET_VARCHAR2 Function Returns and Raised Errors

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>This is the value at the given path position.</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>Raises this error if p_values(p_path) is not an array or object.</td>
</tr>
</tbody>
</table>

Example
This example parses a JSON string and prints the value at a position.

```sql
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_varchar2(p_path=>'items[%d].foo',p0=>3,p_values=>j));
END;
```

21.20 GET_CLOB Function

This function returns clob member value. This function auto-converts varchar2, boolean and number values.

Syntax

```sql
get_clob ( 
    p_path    in varchar2, 
    p0        in varchar2 default null, 
    p1        in varchar2 default null, 
    p2        in varchar2 default null, 
    p3        in varchar2 default null, 
    p4        in varchar2 default null, 
    p_default in clob default null, 
    p_values  in t_values default g_values ) 
return clob
```

Parameters

Table 21-23  GET_CLOB Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_values</td>
<td>Parsed json members. defaults to g_values.</td>
</tr>
<tr>
<td>p_path</td>
<td>Index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path will be replaced by pN and every i-th %s or %d will be replaced by the p[i-1].</td>
</tr>
</tbody>
</table>
### Table 21-23 (Cont.) GET_CLOB Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_default</td>
<td>Default value if the member does not exist.</td>
</tr>
</tbody>
</table>

### Returns/Raised Errors

#### Table 21-24 GET_CLOB Function Returns and Raised Errors

<table>
<thead>
<tr>
<th>Return/Raised Errors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a clob</td>
<td>Value at the given path position</td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>If p_values(p_path) is an array or an object</td>
</tr>
</tbody>
</table>

### Example

Parse a JSON string and print the value at a position.

```sql
declare
    j apex_json.t_values;
begin
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_clob (
        p_values => j,
        p_path => 'items[%d].foo',
        p0 => 3));
end;
```

---

### 21.21 INITIALIZE_CLOB_OUTPUT Procedure

This procedure initializes the output interface to write to a temporary CLOB. The default is to write to SYS.HTP. If using CLOB output, you should call FREE_OUTPUT() at the end to free the CLOB.

#### Syntax

```sql
APEX_JSON.INITIALIZE_CLOB_OUTPUT (  
    p_dur         in pls_integer default sys.dbms_lob.call,  
    p_cache       in boolean     default true,  
    p_indent      in pls_integer default null);
```

#### Parameters

#### Table 21-25 INITIALIZE_CLOB_OUTPUT Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_dur</td>
<td>Duration of the temporary CLOB. this can be DBMS_LOB.SESSION or DBMS_LOB.CALL (the default).</td>
</tr>
</tbody>
</table>
Table 21-25  (Cont.) INITIALIZE_CLOB_OUTPUT Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_cache</td>
<td>Specifies if the lob should be read into buffer cache or not.</td>
</tr>
<tr>
<td>p_indent</td>
<td>Indent level. Defaults to 2 if debug is turned on, 0 otherwise.</td>
</tr>
</tbody>
</table>

Example

This example configures APEX_JSON for CLOB output, generate JSON, print the CLOB with DBMS_OUTPUT, and finally free the CLOB.

```
BEGIN
    apex_json.initialize_clob_output;
    apex_json.open_object;
    apex_json.write('hello', 'world');
    apex_json.close_object;
    dbms_output.put_line(apex_json.get_clob_output);
    apex_json.free_output;
END;
```

21.22 INITIALIZE_OUTPUT Procedure

This procedure initializes the output interface. You only have to call this procedure if you want to modify the parameters below. Initially, output is already configured with the defaults mentioned in the parameter table.

Syntax

APEX_JSON.INITIALIZE_OUTPUT (  
    p_http_header     in boolean     default true,  
    p_http_cache      in boolean     default false,  
    p_http_cache_etag in varchar2    default null,  
    p_indent          in pls_integer default null )

Parameters

Table 21-26  INITIALIZE_OUTPUT Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_http_header</td>
<td>If TRUE (the default), write an application/JSON mime type header.</td>
</tr>
<tr>
<td>p_http_cache</td>
<td>This parameter is only relevant if p_write_header is TRUE. If TRUE, writes Cache-Control: max-age=315360000. If FALSE (the default), writes Cache-Control: no-cache. Otherwise, does not write Cache-Control.</td>
</tr>
</tbody>
</table>
Table 21-26  (Cont.) INITIALIZE_OUTPUT Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>http_cache_etag</td>
<td>If not null, writes an etag header. This parameter is only used if P_HTTP_CACHE is true.</td>
</tr>
<tr>
<td>p_indent</td>
<td>Indent level. Defaults to 2, if debug is turned on, otherwise defaults to 0.</td>
</tr>
</tbody>
</table>

Example

This example configures APEX.JSON to not emit default headers, because they are written directly.

BEGIN
    apex.json.initialize_output (
        p_http_header => false);

    sys.owa_util.mime_header('application/json', false);
    sys.owa_util.status_line(429, 'Too Many Requests');
    sys.owa_util.http_header_close;
    --
    apex.json.open_object;
    apex.json.write('maxRequestsPerSecond', 10);
    apex.json.close_object;
END;

21.23 OPEN_ARRAY Procedure

This procedure writes an open bracket symbol as follows:

[

Syntax

APEX.JSON.OPEN_ARRAY (  
    p_name       IN VARCHAR2 DEFAULT NULL );

Parameters

Table 21-27  OPEN_ARRAY Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>If not null, write an object attribute name and colon before the opening bracket.</td>
</tr>
</tbody>
</table>
Example

This example performs a write { "array": [1, [ ] ] }.

BEGIN
    apex_json.open_object; -- {
    apex_json.open_array(©array©); -- "array": [
    apex_json.write(1); -- 1
    apex_json.open_array; -- , [ 
    apex_json.close_array; -- ]
    apex_json.close_array; -- ]
    apex_json.close_object; -- }
END;

21.24 OPEN_OBJECT Procedure

This procedure writes an open curly bracket symbol as follows:

{ }

Syntax

APEX_JSON.OPEN_OBJECT (p_name IN VARCHAR2 DEFAULT NULL);

Parameters

Table 21-28  OPEN_OBJECT Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>If not null, write an object attribute name and colon before the opening brace.</td>
</tr>
</tbody>
</table>

Example

This example performs a write { "obj": { "obj-attr": "value" } }.

BEGIN
    apex_json.open_object; -- {
    apex_json.open_object('obj'); -- "obj": {
    apex_json.write('obj-attr', 'value'); -- "obj-attr": "value"
    apex_json.close_all; -- }}
END;

21.25 PARSE Procedure Signature 1

This procedure parses a JSON-formatted varchar2 or clob and puts the members into p_values.
Syntax

APEX_JSON.PARSE (  
    p_values   in out nocopy t_values,  
    p_source   in varchar2,  
    p_strict   in boolean default true );

APEX_JSON.PARSE (  
    p_values   in out nocopy t_values,  
    p_source   in clob,  
    p_strict   in boolean default true );

Parameters

Table 21-29  PARSE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_values</td>
<td>An index by varchar2 result array which contains the JSON members and values. The default is g_values.</td>
</tr>
<tr>
<td>p_source</td>
<td>The JSON source (varchar2 or clob)</td>
</tr>
<tr>
<td>p_strict</td>
<td>If TRUE (default), enforce strict JSON rules</td>
</tr>
</tbody>
</table>

Example

This example parses JSON and prints member values.

DECLARE  
    l_values apex_json.t_values;
BEGIN  
    apex_json.parse (  
        p_values => l_values,  
        p_source => '{ "type": "circle", "coord": [10, 20] }' );  
    sys.htp.p('Point at ||
        apex_json.get_number (  
            p_values => l_values,  
            p_path   => 'coord[1]'||
            ','||
        apex_json.get_number (  
            p_values => l_values,  
            p_path => 'coord[2]'));
END;

21.26 PARSE Procedure Signature 2

This procedure parses a JSON-formatted varchar2 or clob and puts the members into the package global g_values. This simplified API works similar to the parse() procedure for signature 1, but saves the developer from declaring a local variable for parsed JSON data and passing it to each JSON API call.
Syntax

APEX_JSON.PARSE (
    p_source   IN VARCHAR2,
    p_strict   IN BOOLEAN DEFAULT TRUE );

APEX_JSON.PARSE (
    p_source   IN CLOB,
    p_strict   IN BOOLEAN DEFAULT TRUE );

Parameters

Table 21-30  PARSE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_source</td>
<td>The JSON source (varchar2 or clob).</td>
</tr>
<tr>
<td>p_strict</td>
<td>If TRUE (default), enforce strict JSON rules.</td>
</tr>
</tbody>
</table>

Example

This example parses JSON and prints member values.

    apex_json.parse('{ "type": "circle", "coord": [10, 20] }');
    sys.htp.p('Point at ' ||
    apex_json.get_number(p_path=>'coord[1]') ||
    ', ' ||
    apex_json.get_number(p_path=>'coord[2]'));

21.27 STRINGIFY Function Signature 1

This function converts a string to an escaped JSON value.

Syntax

APEX_JSON.STRINGIFY (
    p_value   IN VARCHAR2 )
RETURN VARCHAR2;

Parameters

Table 21-31  STRINGIFY Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The string to be converted.</td>
</tr>
</tbody>
</table>
Returns

Table 21-32  STRINGIFY Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>The converted and escaped JSON value.</td>
</tr>
</tbody>
</table>

Example

This example is a query that returns a JSON varchar2 value.

```sql
select apex_json.stringify('line 1' || chr(10) || 'line 2') from dual;
```

21.28 STRINGIFY Function Signature 2

This function converts a number to an escaped JSON value.

Syntax

```sql
APEX_JSON.STRINGIFY (p_value IN NUMBER) RETURN VARCHAR2;
```

Parameters

Table 21-33  STRINGIFY Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The number to be converted.</td>
</tr>
</tbody>
</table>

Returns

Table 21-34  STRINGIFY Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>The converted and escaped JSON value.</td>
</tr>
</tbody>
</table>

Example

This example is a query that returns a JSON number value.

```sql
select apex_json.stringify(-1/10) from dual
```

21.29 STRINGIFY Function Signature 3

This function converts a date to an escaped JSON value.
Syntax

APEX_JSON.STRINGIFY (
    p_value  IN DATE,
    p_format IN VARCHAR2 DEFAULT c_date_iso8601 )
RETURN VARCHAR2;

Parameters

Table 21-35   STRINGIFY Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The date value to be converted.</td>
</tr>
</tbody>
</table>

Returns

Table 21-36   STRINGIFY Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>The converted and escaped JSON value.</td>
</tr>
</tbody>
</table>

Example

This example is a query that returns a JSON varchar2 value that is suitable to be converted to dates.

```
select apex_json.stringify(sysdate) from dual
```

21.30 STRINGIFY Function Signature 4

This function converts a boolean value to an escaped JSON value.

Syntax

APEX_JSON.STRINGIFY (    p_value  IN BOOLEAN )
RETURN VARCHAR2;

Parameters

Table 21-37   STRINGIFY Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The boolean value to be converted.</td>
</tr>
</tbody>
</table>
Returns

Table 21-38 STRINGIFY Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>The converted and escaped JSON value.</td>
</tr>
</tbody>
</table>

Example

This example demonstrates printing JSON boolean values.

BEGIN
    sys.htp.p(apex_json.stringify(true));
    sys.htp.p(apex_json.stringify(false));
END;

21.31 TO_MEMBER_NAME Function

This function converts the given string to a JSON member name, usable for accessing values via the get_% functions. Unless member names are simple identifiers (A-Z, 0-9, ", -"), they need to be quoted.

Syntax

function to_member_name ( 
    p_string in varchar2 )
return varchar2

Parameters

Table 21-39 TO_MEMBER_NAME Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The raw member name.</td>
</tr>
</tbody>
</table>

Returns

A valid member name for get_% functions.

Example

Print various converted strings.

begin
    sys.dbms_output.put_line('Unquoted: ' || apex_json.to_member_name('member_name'));
    sys.dbms_output.put_line('Quoted: ' || apex_json.to_member_name('Hello*World'));
end;
21.32 TO_XMLTYPE Function

This procedure parses a JSON-formatted VARCHAR2 or CLOB and converts it to an xmltype.

Syntax

APEX_JSON.TO_XMLTYPE (
    p_source   IN VARCHAR2,
    p_strict   IN BOOLEAN DEFAULT TRUE )
RETURN sys.xmltype;

APEX_JSON.TO_XMLTYPE (
    p_source   IN CLOB,
    p_strict   IN BOOLEAN DEFAULT TRUE )
RETURN sys.xmltype;

Parameters

Table 21-40    TO_XMLTYPE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_source</td>
<td>The JSON source (VARCHAR2 or CLOB)</td>
</tr>
<tr>
<td>p_strict</td>
<td>If TRUE (default), enforce strict JSON rules</td>
</tr>
</tbody>
</table>

Returns

Table 21-41    TO_XMLTYPE Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys.xmltype</td>
<td>An xmltype representation of the JSON data.</td>
</tr>
</tbody>
</table>

Example

This example parses JSON and prints the XML representation.

DECLARE
    l_xml xmltype;
BEGIN
    l_xml := apex_json.to_xmltype('{} "items": [ 1, 2, { "foo": true }] ');
    dbms_output.put_line(l_xml.getstringval);
END;
21.33 TO_XMLTYPE_SQL Function

This function parses a JSON-formatted VARCHAR2 or CLOB and converts it to an xmltype. This function overload has the p_strict parameter as VARCHAR2 in order to allow invoking from within a SQL query and having JSON parsing in LAX mode.

Syntax

```sql
function to_xmltype_sql ( 
    p_source   IN VARCHAR2, 
    p_strict   IN BOOLEAN DEFAULT 'Y' ) 
RETURN sys.xmltype;
```

```sql
function to_xmltype_sql ( 
    p_source   IN CLOB, 
    p_strict   IN BOOLEAN DEFAULT 'Y' ) 
RETURN sys.xmltype;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_source</td>
<td>The JSON source (VARCHAR2 or CLOB)</td>
</tr>
<tr>
<td>p_strict</td>
<td>If Y (default), enforce strict JSON rules</td>
</tr>
</tbody>
</table>

Returns

An xmltype representation of the json data

Example

This example SQL query converts JSON to XMLTYPE and uses the XMLTABLE SQL function to extract data. The p_strict argument is set to N, so the JSON can successfully be parsed in lax mode, although the items attribute is not enquoted.

```sql
select 
    attr_1 
from 
    xmltable( 
        '/json/items/row' 
        passing apex_json.to_xmltype_sql( '{ items: [ 1, 2, { "foo": true } ] }', p_strict => 'N' ) 
    columns 
    attr_1 varchar2(20) path 'foo/text()' 
    );
```

21.34 WRITE Procedure Signature 1

This procedure writes an array attribute of type VARCHAR2.
Syntax

APEX_JSON.WRITE (  
   p_value    IN VARCHAR2 );

Parameters

Table 21-43    WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The value to be written.</td>
</tr>
</tbody>
</table>

Example

This example writes an array containing 1, "two", "long text", false, the current date and a JSON representation of an xml document.

DECLARE
  l_clob clob := 'long text';
  l_xml sys.xmltype := sys.xmltype('<obj><foo>1</foo><bar>2</bar></obj>');</nBEGIN
  apex_json.open_array; -- [
  apex_json.write(1); -- 1
  apex_json.write('two'); -- , "two"
  apex_json.write(l_clob); -- , "long text"
  apex_json.write(false); -- , false
  apex_json.write(sysdate); -- , "2014-05-05T05:36:08Z"
  apex_json.write(localtimestamp); -- , "2014-05-05T05:36:08.5434Z"
  apex_json.write(current_timestamp); -- , "2014-05-05T05:36:08.5434+02:00"
  apex_json.write(l_xml); -- , { "foo": 1, "bar": 2 }
  apex_json.close_array; -- ]
END;

21.35 WRITE Procedure Signature 2

This procedure writes an array attribute of type clob.

Syntax

APEX_JSON.WRITE (  
   p_value    IN CLOB );

Parameters

Table 21-44    WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The value to be written.</td>
</tr>
</tbody>
</table>
Example
See "WRITE Procedure Signature 1".

21.36 WRITE Procedure Signature 3

This procedure writes an array attribute of type NUMBER.

Syntax

APEX_JSON.WRITE (  
    p_value    IN NUMBER );

Parameters

Table 21-45    WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The value to be written.</td>
</tr>
</tbody>
</table>

Example
See "WRITE Procedure Signature 1".

21.37 WRITE Procedure Signature 4

This procedure writes an array attribute of type date.

Syntax

APEX_JSON.WRITE (  
    p_value    IN DATE,  
    p_format   IN VARCHAR2 DEFAULT c_date_iso8601 );

Parameters

Table 21-46    WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The value to be written.</td>
</tr>
<tr>
<td>p_format</td>
<td>The date format mask (default c_date_iso8601).</td>
</tr>
</tbody>
</table>

Example
See "WRITE Procedure Signature 1".
21.38 WRITE Procedure Signature 5

This procedure writes an array attribute of type boolean.

Syntax

APEX_JSON.WRITE (  
    p_value    IN BOOLEAN );

Parameters

Table 21-47 WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The value to be written.</td>
</tr>
</tbody>
</table>

Example

See "WRITE Procedure Signature 1".

21.39 WRITE Procedure Signature 6

This procedure writes an array attribute of type sys.xmltype. The procedure uses a XSL transformation to generate JSON. To determine the JSON type of values, it uses the following rules:

- If the value is empty, it generates a NULL value.
- If upper(value) is TRUE, it generates a boolean true value.
- If upper(value) is FALSE, it generates a boolean false value.
- If the XPath number function returns TRUE, it emits the value as is. Otherwise, it enquotes the value (that is, treats it as a JSON string).

Syntax

APEX_JSON.WRITE (  
    p_value    IN sys.xmltype );

Parameters

Table 21-48 WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The value to be written.</td>
</tr>
</tbody>
</table>

Example

See "WRITE Procedure Signature 1".
21.40 WRITE Procedure Signature 7

This procedure writes an array with all rows that the cursor returns. Each row is a separate object. If the query contains object type, collection, or cursor columns, the procedure uses `write(xmltype)` to generate JSON. Otherwise, it uses `DBMS_SQL` to fetch rows and the `write()` procedures for the appropriate column data types for output. If the column type is `varchar2` and the uppercase value is 'TRUE' or 'FALSE', it generates boolean values.

Syntax

```sql
APEX_JSON.WRITE (
    p_cursor       IN OUT NOCOPY sys_refcursor);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_cursor</td>
<td>The cursor.</td>
</tr>
</tbody>
</table>

Example 1

This example writes an array containing JSON objects for departments 10 and 20.

```sql
DECLARE
    c sys_refcursor;
BEGIN
    open c for select deptno, dname, loc from dept where deptno in (10, 20);
    apex_json.write(c);
END;
```

This is the output:

```json
[ { "DEPTNO":10 , "DNAME":"ACCOUNTING" , "LOC":"NEW YORK" } , { "DEPTNO":20 , "DNAME":"RESEARCH" , "LOC":"DALLAS" } ]
```

21.41 WRITE Procedure Signature 8

This procedure writes an object attribute of type `VARCHAR2`.

Syntax

```sql
APEX_JSON.WRITE (  
    p_name       IN VARCHAR2,
    p_value      IN VARCHAR2,
    p_write_null IN BOOLEAN  DEFAULT FALSE );
```
## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_value</td>
<td>The attribute value to be written.</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write NULL values. If false (the default), do not write NULLs.</td>
</tr>
</tbody>
</table>

### Example

This example writes an object with named member attributes of various types. The comments to the right of the statements show the output that they generate.

```sql
DECLARE
  l_clob clob := 'long text';
  l_xml sys.xmltype := sys.xmltype('<obj><foo>1</foo><bar>2</bar></obj>');
BEGIN
  apex_json.open_object; -- {  
  apex_json.write('a1', 1); -- "a1": 1
  apex_json.write('a2', 'two'); -- "a2": "two"
  apex_json.write('a3', l_clob); -- "a3": "long text"
  apex_json.write('a4', false); -- "a4": false
  apex_json.write('a5', sysdate); -- "a5": "2014-05-05T05:36:08Z"
  apex_json.write('a6', l_xml); -- "a6": { "foo": 1, "bar": 2 }
  apex_json.close_object; -- }
END;
```

### 21.42 WRITE Procedure Signature 9

This procedure writes an object attribute of type CLOB.

**Syntax**

```sql
APEX_JSON.WRITE (  
  p_name       IN VARCHAR2,  
  p_value      IN CLOB,  
  p_write_null IN BOOLEAN DEFAULT FALSE );
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_value</td>
<td>The attribute value to be written.</td>
</tr>
</tbody>
</table>
21.43 WRITE Procedure Signature 10

This procedure writes an object attribute of type `NUMBER`.

**Syntax**

```sql
APEX_JSON.WRITE (
    p_name         IN VARCHAR2,
    p_value        IN NUMBER,
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_value</td>
<td>The attribute value to be written.</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write NULL values. If false (the default), do not write NULLs.</td>
</tr>
</tbody>
</table>

**Example**

See example for "WRITE Procedure Signature 8".

21.44 WRITE Procedure Signature 11

This procedure writes an object attribute of type `date`.

**Syntax**

```sql
APEX_JSON.WRITE (  
    p_name         IN VARCHAR2,  
    p_value        IN DATE,  
    p_format       IN VARCHAR2 DEFAULT c_date_iso8691,  
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```
### Parameters

#### Table 21-53  WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_value</td>
<td>The attribute value to be written.</td>
</tr>
<tr>
<td>p_format</td>
<td>The date format mask (default apex_json.c_date_iso8601).</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write NULL values. If false (the default), do not write NULL.</td>
</tr>
</tbody>
</table>

#### Example

See example for "WRITE Procedure Signature 8".

### 21.45 WRITE Procedure Signature 12

This procedure writes an object attribute of type boolean.

#### Syntax

```sql
APEX_JSON.WRITE (  
    p_name         IN VARCHAR2,  
    p_value        IN BOOLEAN,  
    p_write_null   IN BOOLEAN  DEFAULT FALSE );
```

#### Parameters

#### Table 21-54  WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_value</td>
<td>The attribute value to be written.</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write NULL values. If false (the default), do not write NULL.</td>
</tr>
</tbody>
</table>

#### Example

See example for "WRITE Procedure Signature 8".

### 21.46 WRITE Procedure Signature 13

This procedure writes an attribute where the value is an array that contains all rows that the cursor returns. Each row is a separate object.

If the query contains object type, collection, or cursor columns, the procedure uses `write(p_name, <xmltype>)`. See "WRITE Procedure Signature 14". Otherwise, it uses `DBMS_SQL` to fetch rows and the `write()` procedures for the appropriate column data types for output. If the column type is `varchar2` and the uppercase value is 'TRUE' or 'FALSE', it generates boolean values.

---

*ORACLE*
Syntax

APEX_JSON.WRITE (  
    p_name IN VARCHAR2,  
    p_cursor IN OUT NOCOPY sys_refcursor );

Parameters

Table 21-55  WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_cursor</td>
<td>The cursor.</td>
</tr>
</tbody>
</table>

Example

This example writes an array containing JSON objects for departments 10 and 20, as an object member attribute.

DECLARE  
c sys_refcursor;
BEGIN  
    open c for select deptno,  
      dname,  
      cursor(select empno,  
        ename  
      from emp e  
      where e.deptno=d.deptno) emps  
    from dept d;  
apex_json.open_object;  
apex_json.write('departments', c);  
apex_json.close_object;
END;

{ "departments":[  
    {"DEPTNO":10,  
      "DNAME":"ACCOUNTING",  
      "EMPS":[{{"EMPNO":7839,"ENAME":"KING"}}],  
      ...  
    },{"DEPTNO":40,"DNAME":"OPERATIONS","EMPS":null} ]

21.47 WRITE Procedure Signature 14

This procedure writes an array attribute of type sys.xmltype. The procedure uses a XSL transformation to generate JSON. To determine the JSON type of values, it uses the following rules:

- If the value is empty, it generates a NULL value.
- If upper(value) is TRUE, it generates a boolean true value.
- If upper(value) is FALSE, it generates a boolean false value.
• If the XPath number function returns true, it emits the value as is. Otherwise, it enquotes the value (that is, treats it as a JSON string).

**Syntax**

APEX_JSON.WRITE (  
    p_name         IN VARCHAR2,  
    p_value        IN sys.xmltype,  
    p_write_null   IN BOOLEAN  DEFAULT FALSE );

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_value</td>
<td>The value to be written. The XML is converted to JSON</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write NULL values. If false (the default), do not write NULLs.</td>
</tr>
</tbody>
</table>

**Example**

See example for "WRITE Procedure Signature 13".

### 21.48 WRITE Procedure Signature 15

This procedure writes parts of a parsed APEX_JSON.t_values table.

**Syntax**

APEX_JSON.WRITE (  
    p_values           IN t_values,  
    p_path             IN VARCHAR2 DEFAULT ©.©,  
    p0                 IN VARCHAR2 DEFAULT NULL,  
    p1                 IN VARCHAR2 DEFAULT NULL,  
    p2                 IN VARCHAR2 DEFAULT NULL,  
    p3                 IN VARCHAR2 DEFAULT NULL,  
    p4                 IN VARCHAR2 DEFAULT NULL );

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_values</td>
<td>The parsed JSON members.</td>
</tr>
<tr>
<td>p_path</td>
<td>The index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path will be replaced by pN and every i-th %s or %d is replaced by p[i-1].</td>
</tr>
</tbody>
</table>
Example

This example parses a JSON string and writes parts of it.

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": { "x": 1, "y": 2 }}');
    apex_json.write(j,'bar');
END;

21.49 WRITE Procedure Signature 16

This procedure writes parts of a parsed APEX_JSON.t_values table as an object member attribute.

Syntax

APEX_JSON.WRITE (  
    p_name           IN VARCHAR2,  
    p_values         IN t_values,  
    p_path           IN VARCHAR2 DEFAULT '','  
    p0               IN VARCHAR2 DEFAULT NULL,  
    p1               IN VARCHAR2 DEFAULT NULL,  
    p2               IN VARCHAR2 DEFAULT NULL,  
    p3               IN VARCHAR2 DEFAULT NULL,  
    p4               IN VARCHAR2 DEFAULT NULL,  
    p_write_null     IN BOOLEAN  DEFAULT FALSE );

Parameters

Table 21-58  WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_values</td>
<td>The parsed JSON members.</td>
</tr>
<tr>
<td>p_path</td>
<td>The index into p_values.</td>
</tr>
<tr>
<td>p[0-4]</td>
<td>Each %N in p_path will be replaced by pN and every i-th %s or %d is replaced by p[i-1].</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write NULL values. If false (the default), do not write NULLs.</td>
</tr>
</tbody>
</table>

Example

This example parses a JSON string and writes parts of it as an object member.

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": { "x": 1, "y": 2 }}');
    apex_json.write(j,'bar');
END;
21.50 WRITE Procedure Signature 17

This procedure writes an array attribute of type VARCHAR2.

Syntax

```sql
procedure write (  
    p_name        in varchar2,  
    p_values      in apex_t_varchar2,  
    p_write_null  in boolean default false );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_values</td>
<td>The VARCHAR2 array values to be written.</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write an empty array. If false (the default), do not -- write an empty array.</td>
</tr>
</tbody>
</table>

Example

This example writes an array containing a, b, c.

```sql
declare  
    l_values apex_t_varchar2 := apex_t_varchar2( 'a', 'b', 'c' );  
begin  
    apex_json.open_object; -- {  
    apex_json.write('array', l_values ); -- "array": [ "a", "b", "c" ]  
    apex_json.close_object; -- }  
end;
```

21.51 WRITE Procedure Signature 18

This procedure writes an array attribute of type NUMBER.

Syntax

```sql
procedure write (  
    p_name        in varchar2,  
    p_values      in apex_t_number,  
    p_write_null  in boolean default false );
```
Parameters

Table 21-60  WRITE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>p_values</td>
<td>The NUMBER array values to be written.</td>
</tr>
<tr>
<td>p_write_null</td>
<td>If true, write an empty array. If false (the default), do not -- write an empty array.</td>
</tr>
</tbody>
</table>

Example

This example writes an array containing 1, 2, 3.

```
declare
    l_values apex_t_number := apex_t_number( 1, 2, 3 );
begin
    apex_json.open_object;  -- {               
    apex_json.write('array', l_values );   -- "array": [ 1, 2, 3 ]
    apex_json.close_object;  -- }             
end;
```
This package provides APIs to work with JSON Web Tokens (JWT). JWTs can be used to pass a number of signed claims between client and server. Token values are URL-safe strings that consist of 3 parts, separated by '.'. The header part identifies the algorithm used for the signature part. The payload part contains the claims to make.

For more details on JWT, see RFC 7519.

- **T_TOKEN**
- **ENCODE Function**
- **DECODE Function**
- **VALIDATE Procedure**

### 22.1 T_TOKEN

A `t_token` record contains the decoded parts of a JSON Web Token.

**Syntax**

```plaintext
type t_token is record (  
    header varchar2(32767),  
    payload varchar2(32767),  
    signature varchar2(32767) );
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>The Javascript Object Signing and Encryption (JOSE) header contains cryptographic parameters.</td>
</tr>
<tr>
<td>payload</td>
<td>The claims which the token asserts.</td>
</tr>
<tr>
<td>signature</td>
<td>The signature of header and payload.</td>
</tr>
</tbody>
</table>

### 22.2 ENCODE Function

This function encodes and optionally encrypts payload.
**Syntax**

```sql
function encode (
    p_iss           in varchar2                 default null,
    p_sub           in varchar2                 default null,
    p_aud           in varchar2                 default null,
    p_nbf_ts        in timestamp with time zone default null,
    p_iat_ts        in timestamp with time zone default systimestamp,
    p_exp_sec       in pls_integer              default null,
    p_jti           in varchar2                 default null,
    p_other_claims  in varchar2                 default null,
    p_signature_key in raw                      default null )
return varchar2
```

**Parameters**

**Table 22-2  ENCODE Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_iss</td>
<td>Optional &quot;iss&quot; (Issuer) claim.</td>
</tr>
<tr>
<td>p_sub</td>
<td>Optional &quot;sub&quot; (Subject) claim.</td>
</tr>
<tr>
<td>p_aud</td>
<td>Optional &quot;aud&quot; (Audience) claim.</td>
</tr>
<tr>
<td>p_nbf_ts</td>
<td>Optional &quot;nbf&quot; (Not Before) claim.</td>
</tr>
<tr>
<td>p_iat_ts</td>
<td>Optional &quot;iat&quot; (Issued At) claim (default systimestamp).</td>
</tr>
<tr>
<td>p_exp_sec</td>
<td>Optional &quot;exp&quot; (Expiration Time) claim, in seconds. The start time is taken</td>
</tr>
<tr>
<td></td>
<td>from &quot;nbf&quot;, &quot;iat&quot; or current time.</td>
</tr>
<tr>
<td>p_jti</td>
<td>Optional &quot;jti&quot; (JWT ID) Claim.</td>
</tr>
<tr>
<td>p_other_claims</td>
<td>Optional raw JSON with additional claims.</td>
</tr>
<tr>
<td>p_signature_key</td>
<td>Optional MAC key for the signature. If not null, a 'HS256' signature is</td>
</tr>
<tr>
<td></td>
<td>added. This requires 12c or higher.</td>
</tr>
</tbody>
</table>

**Returns**

A varchar2, the encoded token value.

**Example**

This example creates and prints a JWT value for Example User, intended to be used by Example JWT Recipient. The token is valid for 5 minutes.

```sql
declare
    l_jwt_value varchar2(32767);
begin
    l_jwt_value := apex_jwt.encode (   
        p_iss => 'Example Issuer',
        p_sub => 'Example User',
        p_aud => 'Example JWT Recipient',
        p_exp_sec => 60*5,
        p_other_claims => '"name1": ''
```
apex_json.stringify('value1') ||
    ', "name2": ' ||
apex_json.stringify('value2'),
    p_signature_key => ... encryption key ... );
sys.dbms_output.put_line(l_jwt_value);
end;

22.3 DECODE Function

This function decodes a raw token value.

Syntax

function decode (  
p_value         in varchar2,
    p_signature_key in raw      default null )
return t_token;

Parameters

Table 22-3  DECODE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>A raw token value contains 3 base64-encoded parts, which are separated by ' .' The parts are header, payload and signature.</td>
</tr>
<tr>
<td>p_signature_key</td>
<td>If not null, validate p_value's signature using this key and the algorithm specified in header. The algorithms 'HS256' and 'none' are supported, but 'HS256' requires 12c or higher.</td>
</tr>
</tbody>
</table>

Returns

A t_token.

Raises

VALUE_ERROR: The input value is invalid.

WWV_FLOW_CRYPTO.UNSUPPORTED_FUNCTION: The token is signed using an unsupported function.

Example

This example decodes an encoded token and print it's contents.

declare
    l_token apex_jwt.t_token;
    l_keys apex_t_varchar2;
begin
    l_token := apex_jwt.decode (  
        p_value =>
            'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9.gzSraSYS8EXBxLN_oWnFSRqCzcmJmMjiuyu5CSpyHI' );
    sys.dbms_output.put_line('--- Header ---');
apex_json.parse(l_token.header);
  l_keys := apex_json.get_members('.');
  for i in 1 .. l_keys.count loop
    sys.dbms_output.put_line(l_keys(i)||'=||
    apex_json.get_varchar2(l_keys(i));
  end loop;
  sys.dbms_output.put_line('--- Payload ---');
  apex_json.parse(l_token.payload);
  l_keys := apex_json.get_members('.');
  for i in 1 .. l_keys.count loop
    sys.dbms_output.put_line(l_keys(i)||'=||
    apex_json.get_varchar2(l_keys(i));
  end loop;
end;

Output:

--- Header ---
alg=HS256
typ=JWT
--- Payload ---
loggedInAs=admin
iat=1422779638

22.4 VALIDATE Procedure

This procedure validates the given token.

Syntax

```sql
procedure validate (    p_token          in t_token,
                        p_iss            in varchar2    default null,
                        p_aud            in varchar2    default null,
                        p_leeway_seconds in pls_integer default 0 );
```

Parameters

Table 22-4 VALIDATE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_token</td>
<td>The JWT.</td>
</tr>
<tr>
<td>p_iss</td>
<td>If not null, verify that the &quot;iss&quot; claim equals p_iss.</td>
</tr>
<tr>
<td>p_aud</td>
<td>If not null, verify that the single &quot;aud&quot; value equals p_aud. If &quot;aud&quot; is an array, verify</td>
</tr>
<tr>
<td></td>
<td>that the &quot;azp&quot; (Authorized Party) claim equals p_aud. This is an OpenID extension.</td>
</tr>
<tr>
<td>p_leeway_seconds</td>
<td>Fudge factor (in seconds) for comparing &quot;exp&quot; (Expiration Time), &quot;nbf&quot; (Not Before) and</td>
</tr>
<tr>
<td></td>
<td>&quot;iat&quot; (Issued At) claims.</td>
</tr>
</tbody>
</table>
 Raises
APEX.ERROR.INTERNAL: Validation failed, check debug log for details.

Example
Verify that l_value is a valid OpenID ID token.

declare
    l_value varchar2(4000) := 'eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso';
    l_oauth2_client_id varchar2(30) := '...';
    l_token apex_jwt.t_token;
begin
    l_token := apex_jwt.decode (p_value => l_value);
    apex_jwt.validate (p_token => l_token,
                        p_aud => l_oauth2_client_id);
end;
You can use APEX_LANG API to translate messages.

- CREATE_LANGUAGE_MAPPING Procedure
- DELETE_LANGUAGE_MAPPING Procedure
- EMIT_LANGUAGE_SELECTOR_LIST Procedure
- LANG Function
- MESSAGE Function
- PUBLISH_APPLICATION Procedure
- SEED_TRANSLATIONS Procedure
- UPDATE_LANGUAGE_MAPPING Procedure
- UPDATE_MESSAGE Procedure
- UPDATE_TRANSLATED_STRING Procedure

### 23.1 CREATE_LANGUAGE_MAPPING Procedure

Use this procedure to create the language mapping for the translation of an application. Translated applications are published as new applications, but are not directly editable in the App Builder.

**Note:**

This procedure is available in Application Express release 4.2.3 and later.

**Syntax**

APEX_LANG.CREATE_LANGUAGE_MAPPING (  
  p_application_id IN NUMBER,  
  p_language IN VARCHAR2,  
  p_translation_application_id IN NUMBER )

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which you want to create the language mapping. This is the ID of the primary language application.</td>
</tr>
</tbody>
</table>
Table 23-1  (Cont.) CREATE_LANGUAGE_MAPPING Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_language</td>
<td>The IANA language code for the mapping. Examples include en-us, fr-ca, ja, he.</td>
</tr>
<tr>
<td>p_translation_application_id</td>
<td>Unique integer value for the ID of the underlying translated application. This number cannot end in 0.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates the creation of the language mapping for an existing Application Express application.

```sql
begin
  --
  -- If running from SQL*Plus, we need to set the environment
  -- for the Application Express workspace associated with this schema.
  The
  -- call to apex_util.set_security_group_id is not necessary if
  -- you're running within the context of the App Builder
  -- or an Application Express application.
  --
  for c1 in (select workspace_id
              from apex_workspaces) loop
    apex_util.set_security_group_id( c1.workspace_id );
    exit;
  end loop;

  -- Now, actually create the language mapping
  apex_lang.create_language_mapping(
    p_application_id => 63969,
    p_language => 'ja',
    p_translation_application_id => 778899 );
  commit;
  --
  -- Print what we just created to confirm
  --
  for c1 in (select *
              from apex_application_trans_map
              where primary_application_id = 63969) loop
    dbms_output.put_line( 'translated_application_id: ' ||
      c1.translated_application_id );
    dbms_output.put_line( 'translated_app_language: ' ||
      c1.translated_app_language );
    end loop;
  end;
/
```

23.2 DELETE_LANGUAGE_MAPPING Procedure

Use this procedure to delete the language mapping for the translation of an application. This procedure deletes all translated strings in the translation repository.
for the specified language and mapping. Translated applications are published as new
applications, but are not directly editable in the App Builder.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

APEX_LANG.DELETE_LANGUAGE_MAPPING (  
   p_application_id IN NUMBER,  
   p_language IN VARCHAR2 )

Parameters

Table 23-2  DELETE_LANGUAGE_MAPPING Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which you want to delete the language mapping. This is the ID of the primary language application.</td>
</tr>
<tr>
<td>p_language</td>
<td>The IANA language code for the existing mapping. Examples include en-us, fr-ca, ja, he.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates the deletion of the language mapping for an existing Application Express application and existing translation mapping.

begin
   --
   -- If running from SQL*Plus, we need to set the environment
   -- for the Application Express workspace associated with this schema.
   The
   -- call to apex_util.set_security_group_id is not necessary if
   -- you're running within the context of the App Builder
   -- or an Application Express application.
   --
   for c1 in (select workspace_id
      from apex_workspaces) loop
      apex_util.set_security_group_id( c1.workspace_id );
      exit;
   end loop;
   -- Now, delete the language mapping
   apex_lang.delete_language_mapping{
      p_application_id => 63969,
      p_language => 'ja' );
   commit;
   --
   -- Print what we just updated to confirm
   --
for c1 in (select count(*) thecount
    from apex_application_trans_map
    where primary_application_id = 63969) loop
    dbms_output.put_line( 'Translation mappings found: ' ||
        c1.thecount );
    end loop;
end; /

23.3 EMIT_LANGUAGE_SELECTOR_LIST Procedure

This procedure determines which languages the current application is translated into and prints language selector. You can use this procedure from a PL/SQL region to include language selector.

Syntax

APEX_LANG.EMIT_LANGUAGE_SELECTOR_LIST;

Example

The following example shows how to use the EMIT_LANGUAGE_SELECTOR_LIST procedure to display language selector.

begin
    APEX_LANG.EMIT_LANGUAGE_SELECTOR_LIST;
end;

23.4 LANG Function

Use this function to return a translated text string for translations defined in dynamic translations.

Syntax

APEX_LANG.LANG (
    p_primary_text_string IN VARCHAR2 DEFAULT NULL,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    ...
    p9 IN VARCHAR2 DEFAULT NULL,
    p_primary_language IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
Parameters

Table 23-3  LANG Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_primary_text_string</td>
<td>Text string of the primary language. This is the value of the Translate From Text in the dynamic translation.</td>
</tr>
<tr>
<td>p0 through p9</td>
<td>Dynamic substitution value: p0 corresponds to %0 in the translation string; p1 corresponds to %1 in the translation string; p2 corresponds to %2 in the translation string, and so on.</td>
</tr>
<tr>
<td>p_primary_language</td>
<td>Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. <strong>See also:</strong> Specifying the Primary Language for an Application in the Oracle Application Express App Builder User's Guide.</td>
</tr>
</tbody>
</table>

Example

Suppose you have a table that defines all primary colors. You could define a dynamic message for each color and then apply the LANG function to the defined values in a query. For example:

```sql
SELECT APEX_LANG.LANG(color)
FROM my_colors
```

If you were running the application in German, RED was a value for the color column in the `my_colors` table, and you defined the German word for red, the previous example would return ROT.

23.5 MESSAGE Function

Use this function to translate text strings (or messages) generated from PL/SQL stored procedures, functions, triggers, packaged procedures, and functions.

Syntax

```sql
APEX_LANG.MESSAGE (p_name            IN VARCHAR2 DEFAULT NULL,
                    p0                IN VARCHAR2 DEFAULT NULL,
                    p1                IN VARCHAR2 DEFAULT NULL,
                    p2                IN VARCHAR2 DEFAULT NULL,
                    ...
                    p9                IN VARCHAR2 DEFAULT NULL,
                    p_lang            IN VARCHAR2 DEFAULT NULL,
                    p_application_id  IN NUMBER DEFAULT NULL)
RETURN VARCHAR2;
```
Parameters

Table 23-4 MESSAGE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the message as defined in Text Messages under Shared Components of your application in Oracle Application Express.</td>
</tr>
<tr>
<td>p0 through p9</td>
<td>Dynamic substitution value: p0 corresponds to %0 in the translation string; p1 corresponds to %1 in the translation string; p2 corresponds to %2 in the translation string, and so on.</td>
</tr>
<tr>
<td>p_lang</td>
<td>Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. See also: Specifying the Primary Language for an Application in the Oracle Application Express App Builder User's Guide.</td>
</tr>
<tr>
<td>p_application_id</td>
<td>Used to specify the application ID within the current workspace that owns the translated message you wish to return. Useful when coding packages that might be called outside of the scope of Oracle Application Express such as packages called from a database job.</td>
</tr>
</tbody>
</table>

Example

The following example assumes you have defined a message called GREETING_MSG in your application in English as "Good morning %0" and in German as "Guten Tag %1". The following example demonstrates how you could invoke this message from PL/SQL:

```plsql
BEGIN
  -- Print the greeting
  --
  HTP.P(APEX_LANG.MESSAGE('GREETING_MSG', V('APP_USER')));
END;
```

How the p_lang attribute is defined depends on how the Application Express engine derives the Application Primary Language. For example, if you are running the application in German and the previous call is made to the APEX_LANG.MESSAGE API, the Application Express engine first looks for a message called GREETING_MSG with a LANG_CODE of de. If it does not find anything, then it is reverted to the Application Primary Language attribute. If it still does not find anything, the Application Express engine looks for a message by this name with a language code of en.

See also:

Specifying the Primary Language for an Application in the Oracle Application Express App Builder User's Guide.
23.6 PUBLISH_APPLICATION Procedure

Use this procedure to publish the translated version of an application. This procedure creates an underlying, hidden replica of the primary application and merges the strings from the translation repository in this new application. Perform a seed and publish process each time you want to update the translated version of your application and synchronize it with the primary application.

This application is not visible in the App Builder. It can be published and exported, but not directly edited.

Note:
This procedure is available in Application Express release 4.2.3 and later.

Syntax

APEX_LANG.PUBLISH_APPLICATION ( p_application_id IN NUMBER, p_language IN VARCHAR2 )

Parameters

Table 23-5  PUBLISH_APPLICATION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which you want to publish and create the translated version. This is the ID of the primary language application.</td>
</tr>
<tr>
<td>p_language</td>
<td>The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates the publish process for an Application Express application and language.

begin
  --
  -- If running from SQL*Plus, we need to set the environment
  -- for the Application Express workspace associated with this schema.
  The call to apex_util.set_security_group_id is not necessary if
  -- you're running within the context of the App Builder
  -- or an Application Express application.
  --
  for c1 in (select workspace_id
              from apex_workspaces) loop
    apex_util.set_security_group_id( c1.workspace_id );

  --
end loop;
-- Now, publish the translated version of the application
apex_lang.publish_application(
    p_application_id => 63969,
    p_language => 'ja');
commit;
end;
/

23.7 SEED_TRANSLATIONS Procedure

Use this procedure to seed the translation repository for the specified application and language. This procedure populates the translation repository with all of the new, updated and removed translatable strings from your application. Perform a seed and publish process each time you want to update the translated version of your application and synchronize it with the primary application.

Syntax

APEX_LANG.SEED_TRANSLATIONS (
    p_application_id IN NUMBER,
    p_language IN VARCHAR2 )

Parameters

Table 23-6  SEED_TRANSLATIONS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which you want to update the translation repository. This is the ID of the primary language application.</td>
</tr>
<tr>
<td>p_language</td>
<td>The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates the seeding process of the translation repository for an Application Express application and language.

begin
--
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema.
    The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for ci in (select workspace_id
            from apex_workspaces) loop
        apex_util.set_security_group_id( ci.workspace_id );
        exit;
end loop;
-- Now, seed the translation repository
apex_lang.seed_translations(
    p_application_id => 63969,
    p_language => 'ja' );
commit;
-- Print out the total number of potentially translatable strings
--
for c1 in (select count(*) thecount
    from apex_application_trans_repos
    where application_id = 63969) loop
    dbms_output.put_line( 'Potentially translatable strings found: ' || c1.thecount );
end loop;
/

23.8 UPDATE_LANGUAGE_MAPPING Procedure

Use this procedure to update the language mapping for the translation of an application. Translated applications are published as new applications, but are not directly editable in the App Builder.

Note:

This procedure is available in Application Express release 4.2.3 and later.

Syntax

APEX_LANG.UPDATE_LANGUAGE_MAPPING (  
p_application_id IN NUMBER,  
p_language IN VARCHAR2,  
p_new_trans_application_id IN NUMBER )

Parameters

Table 23-7  UPDATE_LANGUAGE_MAPPING Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application for which you want to update the language mapping. This is the ID of the primary language application.</td>
</tr>
<tr>
<td>p_language</td>
<td>The IANA language code for the existing mapping. Examples include en-us, fr-ca, ja, he. The language of the mapping cannot be updated with this procedure, only the new translation application ID.</td>
</tr>
<tr>
<td>p_new_trans_application_id</td>
<td>New unique integer value for the ID of the underlying translated application. This number cannot end in 0.</td>
</tr>
</tbody>
</table>
Example

The following example demonstrates the update of the language mapping for an existing Application Express application and existing translation mapping.

```sql
begin
  -- If running from SQL*Plus, we need to set the environment
  -- for the Application Express workspace associated with this schema.
  The
  -- call to apex_util.set_security_group_id is not necessary if
  -- you're running within the context of the App Builder
  -- or an Application Express application.
  --
  for c1 in (select workspace_id
              from apex_workspaces) loop
    apex_util.set_security_group_id( c1.workspace_id );
    exit;
  end loop;
  -- Now, update the language mapping
  apex_lang.update_language_mapping(
    p_application_id => 63969,
    p_language => 'ja',
    p_new_trans_application_id => 881188 );
  commit;
  --
  -- Print what we just updated to confirm
  --
  for c1 in (select *
              from apex_application_trans_map
              where primary_application_id = 63969) loop
    dbms_output.put_line( 'translated_application_id: ' ||
                            c1.translated_application_id );
    dbms_output.put_line( 'translated_app_language: ' ||
                            c1.translated_app_language );
  end loop;
end;
/
```

23.9 UPDATE_MESSAGE Procedure

Use this procedure to update a translatable text message for the specified application.

Note:

This procedure is available in Application Express release 4.2.3 and later.
Syntax

APEX_LANG.UPDATE_MESSAGE ( 
    p_id IN NUMBER,
    p_message_text IN VARCHAR2 )

Parameters

Table 23-8 UPDATE_MESSAGE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_id</td>
<td>The ID of the text message.</td>
</tr>
<tr>
<td>p_message_text</td>
<td>The new text for the translatable text message.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates an update of an existing translatable text message.

begin
    --
    -- If running from SQL*Plus, we need to set the environment
    -- for the Application Express workspace associated with this schema.
    The
    -- call to apex_util.set_security_group_id is not necessary if
    -- you're running within the context of the App Builder
    -- or an Application Express application.
    --
    for c1 in (select workspace_id
                from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Locate the ID of the specific message and update it with the new text
    for c1 in (select translation_entry_id
                from apex_application_translations
                where application_id = 63969
                and translatable_message = 'TOTAL_COST'
                and language_code = 'ja') loop
        apex_lang.update_message( 
            p_id => c1.translation_entry_id,
            p_message_text => 'The total cost is: %0');
        commit;
        exit;
    end loop;
end;
/*
23.10 UPDATE_TRANSLATED_STRING Procedure

Use this procedure to update a translated string in the seeded translation repository.

**Note:**

This procedure is available in Application Express release 4.2.3 and later.

**Syntax**

APEX_LANG.UPDATE_TRANSLATED_STRING (  
    p_id IN NUMBER,  
    p_language IN VARCHAR2  
    p_string IN VARCHAR2 )

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_id</td>
<td>The ID of the string in the translation repository.</td>
</tr>
<tr>
<td>p_language</td>
<td>The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he. The language of the mapping cannot be updated with this procedure, only the new translation application ID.</td>
</tr>
<tr>
<td>p_string</td>
<td>The new value for the string in the translation repository.</td>
</tr>
</tbody>
</table>

**Example**

The following example demonstrates an update of an existing string in the translation repository.

begin  
  -- If running from SQL*Plus, we need to set the environment  
  -- for the Application Express workspace associated with this schema.  
  The  
  -- call to apex_util.set_security_group_id is not necessary if  
  -- you're running within the context of the App Builder  
  -- or an Application Express application.  
  --  
  for cl in (select workspace_id  
                from apex_workspaces) loop  
      apex_util.set_security_group_id( cl.workspace_id );  
      exit;  
  end loop;  
  -- Locate all strings in the repository for the specified application  
  -- which are 'Search' and change to 'Find'
for cl in (select id
    from apex_application_trans_repos
    where application_id = 63969
    and dbms_lob.compare(from_string, to_nctlob('Search')) = 0
    and language_code = 'ja') loop
    apex_lang.update_translated_string(p_id => cl.id,
       p_language => 'ja',
       p_string => 'Find');
    commit;
    exit;
end loop;
end; /
APEX_LDAP

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

- AUTHENTICATE Function
- GET_ALL_USER_ATTRIBUTES Procedure
- GET_USER_ATTRIBUTES Procedure
- IS_MEMBER Function
- MEMBER_OF Function
- MEMBER_OF2 Function
- SEARCH Function

24.1 AUTHENTICATE Function

The AUTHENTICATE function returns a boolean TRUE if the user name and password can be used to perform a SIMPLE_BIND_S, call using the provided search base, host, and port.

Syntax

APEX_LDAP.AUTHENTICATE(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_password     IN VARCHAR2 DEFAULT NULL,
    p_search_base  IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN BOOLEAN;

Parameters

Table 24-1  AUTHENTICATE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Login name of the user.</td>
</tr>
<tr>
<td>p_password</td>
<td>Password for p_username.</td>
</tr>
<tr>
<td>p_search_base</td>
<td>LDAP search base, for example, dc=users, dc=my, dc=org.</td>
</tr>
<tr>
<td>p_host</td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td>p_port</td>
<td>LDAP server port number.</td>
</tr>
</tbody>
</table>
Table 24-1  (Cont.) AUTHENTICATE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_use_ssl</td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the APEX_LDAP.AUTHENTICATE function to verify user credentials against an LDAP Server.

```sql
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_server.my_company.com',
    p_port => 389) THEN
    dbms_output.put_line('authenticated');
ELSE
    dbms_output.put_line('authentication failed');
END IF;
```

24.2 GET_ALL_USER_ATTRIBUTES Procedure

The GET_ALL_USER_ATTRIBUTES procedure returns two OUT arrays of user_attribute names and values for the user name designated by p_username (with password if required) using the provided auth base, host, and port.

Syntax

```sql
APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2 DEFAULT NULL,
    p_host              IN VARCHAR2,
    p_port              IN VARCHAR2 DEFAULT 389,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
    p_attributes        OUT apex_application_global.vc_arr2,
    p_attribute_values  OUT apex_application_global.vc_arr2);
```

Parameters

Table 24-2  GET_ALL_USER_ATTRIBUTES Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Login name of the user.</td>
</tr>
<tr>
<td>p_pass</td>
<td>Password for p_username.</td>
</tr>
</tbody>
</table>
### Table 24-2  (Cont.) GET_ALL_USER_ATTRIBUTES Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_auth_base</td>
<td>LDAP search base, for example, dc=users,dc=my,dc=org.</td>
</tr>
<tr>
<td>p_host</td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td>p_port</td>
<td>LDAP server port number.</td>
</tr>
<tr>
<td>p_use_ssl</td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).</td>
</tr>
<tr>
<td>p_attributes</td>
<td>An array of attribute names returned.</td>
</tr>
<tr>
<td>p_attribute_values</td>
<td>An array of values returned for each corresponding attribute name returned in p_attributes.</td>
</tr>
</tbody>
</table>

### Example

The following example demonstrates how to use the APEX_LDAP.GET_ALL_USER_ATTRIBUTES procedure to retrieve all attribute value's associated to a user.

```sql
DECLARE
    L_ATTRIBUTES apex_application_global.vc_arr2;
    L_ATTRIBUTE_VALUES apex_application_global.vc_arr2;
BEGIN
    APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
        p_username         => 'firstname.lastname',
        p_pass             => 'abcdef',
        p_auth_base        => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host             => 'our_ldap_server.my_company.com',
        p_port             => '389',
        p_attributes       => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);
    FOR i IN L_ATTRIBUTES.FIRST..L_ATTRIBUTES.LAST LOOP
        htp.p('attribute name: '||L_ATTRIBUTES(i));
        htp.p('attribute value: '||L_ATTRIBUTE_VALUES(i));
    END LOOP;
END;
```

### 24.3 GET_USER_ATTRIBUTES Procedure

The GET_USER_ATTRIBUTES procedure returns an OUT array of user_attribute values for the user name designated by p_username (with password if required) corresponding to the attribute names passed in p_attributes using the provided auth base, host, and port.

**Syntax**

```sql
APEX_LDAP.GET_USER_ATTRIBUTES(
    p_username IN VARCHAR2 DEFAULT NULL,
    p_pass IN VARCHAR2 DEFAULT NULL,
);```
Parameters

Table 24-3  GET_USER_ATTRIBUTES Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Login name of the user.</td>
</tr>
<tr>
<td>p_pass</td>
<td>Password for p_username.</td>
</tr>
<tr>
<td>p_auth_base</td>
<td>LDAP search base, for example, dc=users,dc=my,dc=org.</td>
</tr>
<tr>
<td>p_host</td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td>p_port</td>
<td>LDAP server port number.</td>
</tr>
<tr>
<td>p_use_ssl</td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).</td>
</tr>
<tr>
<td>p_attributes</td>
<td>An array of attribute names for which values are to be returned.</td>
</tr>
<tr>
<td>p_attribute_values</td>
<td>An array of values returned for each corresponding attribute name in p_attributes.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the APEX_LDAP.GET_USER_ATTRIBUTES procedure to retrieve a specific attribute value associated to a user.

```sql
DECLARE
    L_ATTRIBUTES apex_application_global.vc_arr2;
    L_ATTRIBUTE_VALUES apex_application_global.vc_arr2;
BEGIN
    L_ATTRIBUTES(1) := 'xxxxxxxxxx'; /* name of the employee number attribute */
    APEX_LDAP.GET_USER_ATTRIBUTES(
        p_username => 'firstname.lastname',
        p_pass => NULL,
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_server.my_company.com',
        p_port => '389',
        p_attributes => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);
END;
```
24.4 IS_MEMBER Function

The IS_MEMBER function returns a boolean TRUE if the user named by p_username (with password if required) is a member of the group specified by the p_group and p_group_base parameters using the provided auth base, host, and port.

Syntax

APEX_LDAP.IS_MEMBER(
    p_username     IN VARCHAR2,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N',
    p_group        IN VARCHAR2,
    p_group_base   IN VARCHAR2)
RETURN BOOLEAN;

Parameters

Table 24-4 IS_MEMBER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Login name of the user.</td>
</tr>
<tr>
<td>p_pass</td>
<td>Password for p_username.</td>
</tr>
<tr>
<td>p_auth_base</td>
<td>LDAP search base, for example, dc=users,dc=my,dc=org.</td>
</tr>
<tr>
<td>p_host</td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td>p_port</td>
<td>LDAP server port number.</td>
</tr>
<tr>
<td>p_use_ssl</td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL.</td>
</tr>
<tr>
<td>p_group</td>
<td>Name of the group to be search for membership.</td>
</tr>
<tr>
<td>p_group_base</td>
<td>The base from which the search should be started.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the APEX_LDAP.IS_MEMBER function to verify whether a user is a member of a group against an LDAP server.

DECLARE
    L_VAL boolean;
BEGIN
    L_VAL := APEX_LDAP.IS_MEMBER(
        p_username => 'firstname.lastname',
        p_pass => 'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_server.my_company.com',
        p_port => 389,
    );
The MEMBER_OF function returns an array of groups the user name designated by p_username (with password if required) belongs to, using the provided auth base, host, and port.

### Syntax

APEX_LDAP.MEMBER_OF(
    p_username IN VARCHAR2 DEFAULT NULL,
    p_pass IN VARCHAR2 DEFAULT NULL,
    p_auth_base IN VARCHAR2,
    p_host IN VARCHAR2,
    p_port IN VARCHAR2 DEFAULT 389,
    p_use_ssl IN VARCHAR2 DEFAULT 'N')
RETURN apex_application_global.vc_arr2;

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Login name of the user.</td>
</tr>
<tr>
<td>p_pass</td>
<td>Password for p_username.</td>
</tr>
<tr>
<td>p_auth_base</td>
<td>LDAP search base, for example, dc=users,dc=my,dc=org.</td>
</tr>
<tr>
<td>p_host</td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td>p_port</td>
<td>LDAP server port number.</td>
</tr>
<tr>
<td>p_use_ssl</td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).</td>
</tr>
</tbody>
</table>

### Example

The following example demonstrates how to use the APEX_LDAP.MEMBER_OF function to retrieve all the groups designated by the specified username.

```sql
DECLARE
    L_MEMBERSHIP apex_application_global.vc_arr2;
BEGIN
    L_MEMBERSHIP := APEX_LDAP.MEMBER_OF(
        p_username => 'firstname.lastname',
        p_group => 'group_name',
        p_group_base => 'group_base');
    IF L_VAL THEN
        htp.p('Is a member.');
    ELSE
        htp.p('Not a member.');
    END IF;
END;
```
24.6 MEMBER_OF2 Function

The **MEMBER_OF2** function returns a VARCHAR2 colon delimited list of groups the user name designated by `p_username` (with password if required) belongs to, using the provided auth base, host, and port.

**Syntax**

```
APEX_LDAP.MEMBER_OF2(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN VARCHAR2;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_username</code></td>
<td>Login name of the user.</td>
</tr>
<tr>
<td><code>p_pass</code></td>
<td>Password for <code>p_username</code>.</td>
</tr>
<tr>
<td><code>p_auth_base</code></td>
<td>LDAP search base, for example, <code>dc=users,dc=my,dc=org</code>.</td>
</tr>
<tr>
<td><code>p_host</code></td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td><code>p_port</code></td>
<td>LDAP server port number.</td>
</tr>
<tr>
<td><code>p_use_ssl</code></td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).</td>
</tr>
</tbody>
</table>

**Example**

The following example demonstrates how to use the **APEX_LDAP.MEMBER_OF2** function to retrieve all the groups designated by the specified username.

```
DECLARE
    L_VAL varchar2(4000);
BEGIN
    L_VAL := APEX_LDAP.MEMBER_OF2(
        p_username => 'firstname.lastname',
        p_pass => 'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => 389);
    FOR i IN L_MEMBERSHIP.FIRST..L_MEMBERSHIP.LAST LOOP
        htp.p('Member of: ||L_MEMBERSHIP(i));
    END LOOP;
END;
```
24.7 SEARCH Function

The `SEARCH` function searches the LDAP repository. The result is an object table of (dn, name, val) that can be used in table queries.

**Syntax**

```sql
function search (    
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2 DEFAULT NULL,
    p_host              IN VARCHAR2,
    p_port              IN NUMBER DEFAULT 389,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
    p_search_base       IN VARCHAR2,
    p_search_filter     IN VARCHAR2,
    p_scope             IN BINARY_INTEGER DEFAULT SYS.DBMS_LDAP.SCOPE_SUBTREE,
    p_timeout_sec       IN BINARY_INTEGER DEFAULT 3,
    p_attribute_names   IN VARCHAR2 ) 
RETURN APEX_T_LDAP_ATTRIBUTES PIPELINED;
```

**Parameters**

**Table 24-7 Search Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Login name of the user (can be NULL for anonymous binds).</td>
</tr>
<tr>
<td>p_pass</td>
<td>The password for <code>p_username</code> (can be NULL for anonymous binds).</td>
</tr>
<tr>
<td>p_auth_base</td>
<td>The authentication base dn for <code>p_username</code> (for example, dc=users,dc=my,dc=org). Can be NULL for anonymous binds.</td>
</tr>
<tr>
<td>p_host</td>
<td>The LDAP server host name.</td>
</tr>
<tr>
<td>p_port</td>
<td>The LDAP server port number.</td>
</tr>
<tr>
<td>p_use_ssl</td>
<td>Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).</td>
</tr>
<tr>
<td>p_search_base</td>
<td>dn base for the search.</td>
</tr>
<tr>
<td>p_search_filter</td>
<td>LDAP search filter expression.</td>
</tr>
<tr>
<td>p_scope</td>
<td>Search scope (default descends into subtrees).</td>
</tr>
</tbody>
</table>
Table 24-7  (Cont.) Search Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_timeout_sec</td>
<td>Timeout for the search (default is 3 seconds)</td>
</tr>
<tr>
<td>p_attribute_names</td>
<td>Comma separated list of return attribute names</td>
</tr>
</tbody>
</table>

Example 1

```
SELECT val group_dns
    FROM table(apex_ldap.search (  
        p_host => 'ldap.example.com',
        p_search_base => 'dc=example,dc=com',
        p_search_filter => 'uid=||
        apex_escape.ldap_search_filter(:APP_USER),
        p_attribute_names => 'memberof'));
```

Example 2

```
SELECT dn, mail, dispname, phone
    from ( select dn, name, val
        from table(apex_ldap.search (  
            p_host => 'ldap.example.com',
            p_search_base => 'dc=example,dc=com',
            p_search_filter => '(objectClass=person)'
            ou=Test'),
            p_attribute_names =>
            'mail,displayname,telephonenumber' ))
    pivot (min(val) for name in ( 'mail' => mail,
                               'displayname' => dispname,
                               'telephonenumber' => phone ))
```
You can use the APEX_MAIL package to send an email from an Oracle Application Express application. This package is built on top of the Oracle supplied UTL_SMTP package. Because of this dependence, the UTL_SMTP package must be installed and functioning to use APEX_MAIL.

APEX_MAIL contains three procedures. Use APEX_MAIL.SEND to send an outbound email message from your application. Use APEX_MAIL.PUSH_QUEUE to deliver mail messages stored in APEX_MAIL_QUEUE. Use APEX_MAIL.ADD_ATTACHMENT to send an outbound email message from your application as an attachment.

Oracle Application Express installs the database job ORACLE_APEX_MAIL_QUEUE, which periodically sends all mail messages stored in the active mail queue.

**Note:**

In order to call the APEX_MAIL package from outside the context of an Application Express application (such as from a Database Scheduler job), you must call apex_util.set_security_group_id as in the following example:

```sql
for c1 in (select workspace_id
            from apex_applications
            where application_id = p_app_id )
loop
    apex_util.set_security_group_id(p_security_group_id =>
    c1.workspace_id);
end loop;
```

- Configuring Oracle Application Express to Send Email
- ADD_ATTACHMENT Procedure
- GET_IMAGES_URL Function
- GET_INSTANCE_URL Function
- PREPARE_TEMPLATE Procedure
- PUSH_QUEUE Procedure
- SEND Procedure
- SEND Function
- SEND Procedure
- SEND Function
25.1 Configuring Oracle Application Express to Send Email

Before you can send email from an App Builder application, you must:

1. Log in to Oracle Application Express Administration Services and configure the email settings on the Instance Settings page. See "Configuring Email" in Oracle Application Express Administration Guide.

2. If you are running Oracle Application Express with Oracle Database 11g release 1 (11.1), you must enable outbound mail. In Oracle Database 11g release 1 (11.1), the ability to interact with network services is disabled by default. See “Enabling Network Service in Oracle Database 11g” in Oracle Application Express App Builder User’s Guide.

Tip:
You can configure Oracle Application Express to automatically email users their login credentials when a new workspace request has been approved. To learn more, see "Selecting a Provisioning Mode" in Oracle Application Express Administration Guide.

25.2 ADD_ATTACHMENT Procedure

This procedure sends an outbound email message from an application as an attachment. To add multiple attachments to a single email, APEX_MAIL.ADD_ATTACHMENT can be called repeatedly for a single email message.

Syntax

APEX_MAIL.ADD_ATTACHMENT(
    p_mail_id IN NUMBER,
    p_attachment IN BLOB,
    p_filename IN VARCHAR2,
    p_mime_type IN VARCHAR2);
Parameters

Table 25-1  ADD_ATTACHMENT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_mail_id</td>
<td>The numeric ID associated with the email. This is the numeric identifier returned from the call to APEX_MAIL.SEND to compose the email body.</td>
</tr>
<tr>
<td>p_attachment</td>
<td>A BLOB variable containing the binary content to be attached to the email message.</td>
</tr>
<tr>
<td>p_filename</td>
<td>The filename associated with the email attachment.</td>
</tr>
<tr>
<td>p_mime_type</td>
<td>A valid MIME type (or Internet media type) to associate with the email attachment.</td>
</tr>
</tbody>
</table>

Examples

The following example demonstrates how to access files stored in APEX_APPLICATION_FILES and add them to an outbound email message.

```
DECLARE
    l_id NUMBER;
BEGIN
    l_id := APEX_MAIL.SEND(
        p_to => 'fred@flintstone.com',
        p_from => 'barney@rubble.com',
        p_subj => 'APEX_MAIL with attachment',
        p_body => 'Please review the attachment.',
        p_body_html => '<b>Please</b> review the attachment');
    FOR c1 IN (SELECT filename, blob_content, mime_type
               FROM APEX_APPLICATION_FILES
               WHERE ID IN (123,456)) LOOP
        APEX_MAIL.ADD_ATTACHMENT(
            p_mail_id => l_id,
            p_attachment => c1.blob_content,
            p_filename => c1.filename,
            p_mime_type => c1.mime_type);
    END LOOP;
    COMMIT;
END;
/
```

25.3 GET_IMAGES_URL Function

Use this function to get the image prefixed URL, if the email includes Application Express instance images.

Syntax

```
APEX_MAIL.GET_IMAGES_URL return VARCHAR2;
```
Parameters
None.

Example
The following example sends an Order Confirmation email which includes the Oracle Logo image.

```sql
declare
    l_body      clob;
    l_body_html clob;
begin
    l_body := 'To view the content of this message, please use an HTML enabled mail client.' || utl_tcp.crlf;

    l_body_html := '<html><body>' || utl_tcp.crlf || '
    <p>Please confirm your order on the <a href="" || apex_mail.get_instance_url || 'f?p=100:10">Order Confirmation</a> page.</p>
    <p>Sincerely,<br />
    The Application Express Dev Team<br />
    <img src="" || apex_mail.get_images_url || 'oracle.gif" alt="Oracle Logo"></p>
    '<body></html>'; \n
    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      => 'Order Confirmation');
end;
```

25.4 GET_INSTANCE_URL Function

If an email includes a link to an Application Express instance, use this function to get the instance URL.

**Note:**

This function requires that the instance setting Application Express Instance URL for emails is set.

Syntax

```
APEX_MAIL.GET_INSTANCE_URL return VARCHAR2;
```
Parameters
None.

Example
The following example sends an Order Confirmation email which includes an absolute URL to page 10 of application 100.

```
declare
    l_body      clob;
    l_body_html clob;
begin
    l_body := 'To view the content of this message, please use an HTML enabled mail client.' || utl_tcp.crlf;
    l_body := '<html><body>' || utl_tcp.crlf ||
              '<p>Please confirm your order on the <a href="' ||
              apex_mail.get_instance_url || '" f?p=100:10">Order Confirmation</a> page.</p>' || utl_tcp.crlf ||
              '</body></html>';
    apex_mail.send (p_to        => 'some_user@somewhere.com', -- change to your email address
                     p_from     => 'some_sender@somewhere.com', -- change to a real senders email address
                     p_body     => l_body,
                     p_body_html => l_body_html,
                     p_subj     => 'Order Confirmation' );
end;
```

25.5 PREPARE_TEMPLATE Procedure

Procedure to return a formatted mail based on an e-mail template where the placeholders specified as json string are substituted.

Syntax

```
PROCEDURE PREPARE_TEMPLATE (  
P_STATIC_ID       IN      VARCHAR2,  
P_PLACEHOLDERS    IN     CLOB,  
P_APPLICATION_ID  IN     NUMBER DEFAULT,  
P_SUBJECT         OUT    VARCHAR2,  
P_HTML            OUT    CLOB,  
P_TEXT             OUT    CLOB );
```
Parameters

Table 25-2 PREPARE_TEMPLATE Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_static_id</td>
<td></td>
</tr>
<tr>
<td>p_placeholders</td>
<td></td>
</tr>
<tr>
<td>p_application_id</td>
<td></td>
</tr>
<tr>
<td>p_subject</td>
<td></td>
</tr>
<tr>
<td>p_html</td>
<td></td>
</tr>
<tr>
<td>p_text</td>
<td></td>
</tr>
</tbody>
</table>

Example

```
declare
    l_subject      varchar2( 4000 );
    l_html         clob;
    l_text         clob;
begin
    apex_mail.prepare_template (p_static_id    => 'ORDER',
                                p_placeholders => '{ "ORDER_NUMBER": 5321, "ORDER_DATE": "01-Feb-2018", "ORDER_TOTAL": "$12,000" }',
                                p_subject      => l_subject,
                                p_html         => l_html,
                                p_text         => l_text);
end;
```

25.6 PUSH_QUEUE Procedure

Oracle Application Express stores unsent email messages in a table named APEX_MAIL_QUEUE. You can manually deliver mail messages stored in this queue to the specified SMTP gateway by invoking the APEX_MAIL.PUSH_QUEUE procedure.

Oracle Application Express logs successfully submitted message in the table APEX_MAIL_LOG with the timestamp reflecting your server’s local time.

Syntax

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

Table 25-3  PUSH_QUEUE Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_smtp_hostname</td>
<td>SMTP gateway host name</td>
</tr>
<tr>
<td>p_smtp_portno</td>
<td>SMTP gateway port number</td>
</tr>
</tbody>
</table>

Note that these parameter values are provided for backward compatibility, but their respective values are ignored. The SMTP gateway hostname and SMTP gateway port number are exclusively derived from values entered on the Manage Environment Settings when sending email.

Example

The following example demonstrates the use of the APEX_MAIL.PUSH_QUEUE procedure using a shell script. This example only applies to UNIX/LINUX installations.

```sql
SQLPLUS / <<EOF
APEX_MAIL.PUSH_QUEUE;
DISCONNECT
EXIT
EOF
```

See Also:

- “Sending an Email from an Application” in Oracle Application Express App Builder User’s Guide
- “Configuring Email Settings” in Oracle Application Express Administration Guide
- “Sending Email from an Application” in Oracle Application Express App Builder User’s Guide

25.7 SEND Procedure

This procedure is used to add a mail to the mail queue of Application Express. The mail is based on an e-mail template where the placeholder values specified as json string are substituted.

Syntax

```sql
PROCEDURE SEND (  
    P_TEMPLATE_STATIC_ID IN VARCHAR2,  
    P_PLACEHOLDERS      IN CLOB,  
    p_to                IN VARCHAR2,  
    p_cc                IN VARCHAR2 DEFAULT NULL,  
```
p_bcc                  IN VARCHAR2 DEFAULT NULL,
p_from                 IN VARCHAR2 DEFAULT NULL,
p_replyto              IN VARCHAR2 DEFAULT NULL,
p_application_id       IN NUMBER   DEFAULT wwv_flow_security.g_flow_id );

Parameters

Table 25-4    SEND Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_template_static_id</td>
<td>Static identifier string, used to identify the shared component email template.</td>
</tr>
<tr>
<td>p_placeholders</td>
<td>JSON string representing the placeholder names along with the values, to be substituted.</td>
</tr>
<tr>
<td>p_to</td>
<td>Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list.</td>
</tr>
<tr>
<td>p_cc</td>
<td>Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list.</td>
</tr>
<tr>
<td>p_bcc</td>
<td>Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list.</td>
</tr>
<tr>
<td>p_from</td>
<td>Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent.</td>
</tr>
<tr>
<td>p_replyto</td>
<td>Address of the Reply-To mail header. You can use this parameter as follows:</td>
</tr>
<tr>
<td></td>
<td>• If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter</td>
</tr>
<tr>
<td></td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td>• If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you send these messages, but the automatic replies go to the value specified (for example, the email address)</td>
</tr>
<tr>
<td>p_application_id</td>
<td>Application ID where the email template is defined. Defaults to the current application (if called from within an application).</td>
</tr>
</tbody>
</table>

Examples

begin
  apex_mail.send {
    p_template_static_id => 'ORDER',
    p_placeholders       => '{ "ORDER_NUMBER": 5321, "ORDER_DATE": "01-Feb-2018", "ORDER_TOTAL": "$12,000" }',
    p_to                 => 'some_user@somewhere.com' ;
  }
end;
25.8 SEND Function

This function returns a mail id after adding the mail to the mail queue of Application Express. The mail id can be used in a call to `add_attachment` to add attachments to an existing mail.

The mail is based on an e-mail template where the placeholder values specified as json string are substituted.

Syntax

```sql
FUNCTION SEND ( 
    P_TEMPLATE_STATIC_ID IN VARCHAR2,
    P_PLACEHOLDERS IN CLOB,
    P_TO IN VARCHAR2,
    P_CC IN VARCHAR2 DEFAULT NULL,
    P_BCC IN VARCHAR2 DEFAULT NULL,
    P_FROM IN VARCHAR2 DEFAULT NULL,
    P_REPLYTO IN VARCHAR2 DEFAULT NULL,
    PAPPLICATION_ID IN NUMBER DEFAULT wwv_flow_security.g_flow_id )
RETURN NUMBER;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_template_static_id</td>
<td>Static identifier string, used to identify the shared component email template.</td>
</tr>
<tr>
<td>p_placeholders</td>
<td>JSON string representing the placeholder names along with the values, to be substituted.</td>
</tr>
<tr>
<td>p_to</td>
<td>Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list.</td>
</tr>
<tr>
<td>p_cc</td>
<td>Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list.</td>
</tr>
<tr>
<td>p_bcc</td>
<td>Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list.</td>
</tr>
<tr>
<td>p_from</td>
<td>Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent.</td>
</tr>
<tr>
<td>p_replyto</td>
<td>Address of the Reply-To mail header. You can use this parameter as follows:</td>
</tr>
<tr>
<td></td>
<td>• If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter</td>
</tr>
<tr>
<td></td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td>• If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you send these messages, but the automatic replies go to the value specified (for example, the email address)</td>
</tr>
</tbody>
</table>
Table 25-5  (Cont.) SEND Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>Application ID where the email template is defined. Defaults to the current application (if called from within an application).</td>
</tr>
</tbody>
</table>

Examples

```sql
declare
    l_mail_id number;
begin
    l_mail_id := apex_mail.send (
        p_template_static_id => 'ORDER',
        p_placeholders       => '{ "ORDER_NUMBER": 5321, "ORDER_DATE": "01-Feb-2018", "ORDER_TOTAL": "$12,000" }',
        p_to                 => 'some_user@somewhere.com' );

    apex_mail.add_attachment ( p_mail_id => l_mail_id, p_attachment => ... );
end;
```

25.9 SEND Procedure

This procedure sends an outbound email message from an application. Although you can use this procedure to pass in either a VARCHAR2 or a CLOB to p_body and p_body_html, the data types must be the same. In other words, you cannot pass a CLOB to P_BODY and a VARCHAR2 to p_body_html.

When using APEX_MAIL.SEND, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your p_body or p_body_html parameters into chunks of 1000 characters or less. Failing to do so results in erroneous email messages, including partial messages or messages with extraneous exclamation points.

- **Plain text and HTML email content.** Passing a value to p_body, but not p_body_html results in a plain text message. Passing a value to p_body and p_body_html yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient’s email client determine what displays. Although most modern email clients can read an HTML formatted email, remember that some users disable this functionality to address security issues.

- **Avoid images.** When referencing images in p_body_html using the `<img />` tag, remember that the images must be accessible to the recipient’s email client in order for them to see the image.

For example, suppose you reference an image on your network called hello.gif as follows:

```html
<img src="http://someserver.com/hello.gif" alt="Hello" />
```
In this example, the image is not attached to the email, but is referenced by the email. For the recipient to see it, they must be able to access the image using a web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image is not displayed. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```sql
APEX_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
    p_body_html                 IN  [ VARCHAR2 | CLOB ] DEFAULT NULL,
    p_subj                      IN    VARCHAR2 DEFAULT NULL,
    p_cc                        IN    VARCHAR2 DEFAULT NULL,
    p_bcc                       IN    VARCHAR2 DEFAULT NULL,
    p_replyto                   IN    VARCHAR2);
```

Parameters

Table 25-6  SEND Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_to</td>
<td>Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list</td>
</tr>
<tr>
<td>p_from</td>
<td>Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent</td>
</tr>
<tr>
<td>p_body</td>
<td>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.</td>
</tr>
<tr>
<td>p_body_html</td>
<td>Body of the email in HTML format. This must be a full HTML document including the &lt;html&gt; and &lt;body&gt; tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)</td>
</tr>
<tr>
<td>p_subj</td>
<td>Subject of the email</td>
</tr>
<tr>
<td>p_cc</td>
<td>Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list</td>
</tr>
<tr>
<td>p_bcc</td>
<td>Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list</td>
</tr>
</tbody>
</table>
Table 25-6  (Cont.) SEND Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| p_replyto | Address of the Reply-To mail header. You can use this parameter as follows:  
• If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter  
• If you include the p_replyto parameter, but provide a NULL value, the Reply-To mail header is set to NULL. This results in the suppression of automatic email replies  
• If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you send these messages, but the automatic replies go to the value specified (for example, the email address) |

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application.

```sql
-- Example One: Plain Text only message
DECLARE
  l_body      CLOB;
BEGIN
  l_body := ©Thank you for your interest in the APEX_MAIL package.©||utl_tcp.crlf||utl_tcp.crlf;
  l_body := l_body ||©  Sincerely,©||utl_tcp.crlf;
  l_body := l_body ||©  The Application Express Dev Team©||utl_tcp.crlf;
  apex_mail.send(
    p_to       => ©some_user@somewhere.com©,   -- change to your email address
    p_from     => ©some_sender@somewhere.com©, -- change to a real senders email address
    p_body     => l_body,
    p_subj     => ©APEX_MAIL Package - Plain Text message©);
END;
/
```

The following example demonstrates how to use APEX_MAIL.SEND to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses utl_tcp.crlf.

```sql
-- 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 := l_body ||'©<html>
  <head>
  <title>APEX_MAIL Package - HTML message</title>
  </head>
  <body>
  To view the content of this message, please use an HTML enabled mail client.©'©||utl_tcp.crlf;
  l_body_html := '<html>
    <head>
    <title>APEX_MAIL Package - HTML message</title>
    </head>
    <body>
    To view the content of this message, please use an HTML enabled mail client.©
    </body>
    </html>
  apex_mail.send(
    p_to       => 'some_user@somewhere.com',   -- change to your email address
    p_from     => 'some_sender@somewhere.com', -- change to a real senders email address
    p_body     => l_body,
    p_subj     => 'APEX_MAIL Package - Plain Text & HTML message');
END;
/
```
25.10 SEND Function

This function sends an outbound email message from an application. Although you can use this function to pass in either a VARCHAR2 or a CLOB to p_body and p_body_html, the data types must be the same. In other words, you cannot pass a CLOB to p_body and a VARCHAR2 to p_body_html.

This function returns a NUMBER. The NUMBER returned is the unique numeric identifier associated with the mail message.

When using APEX_MAIL.send, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your p_body or p_body_html parameters into chunks of 1000 characters or less. Failing to do so results in erroneous email messages, including partial messages or messages with extraneous exclamation points.

- **Plain text and HTML email content.** Passing a value to p_body, but not p_body_html results in a plain text message. Passing a value to p_body and p_body_html yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient's email client determine what displays. Although most modern email clients can read an HTML formatted email, remember that some users disable this functionality to address security issues.
• Avoid images. When referencing images in `p_body_html` using the `<img />` tag, remember that the images must be accessible to the recipient’s email client in order for them to see the image.

For example, suppose you reference an image on your network called `hello.gif` as follows:

```html
<img src="http://someserver.com/hello.gif" alt="Hello" />
```

In this example, the image is not attached to the email, but is referenced by the email. For the recipient to see it, they must be able to access the image using a web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image is not displayed. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```sql
APEX_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
    p_body_html                 IN  [ VARCHAR2 | CLOB ] DEFAULT NULL,
    p_subj                      IN    VARCHAR2 DEFAULT NULL,
    p_cc                        IN    VARCHAR2 DEFAULT NULL,
    p_bcc                       IN    VARCHAR2 DEFAULT NULL,
    p_replyto                   IN    VARCHAR2)
return NUMBER;
```

Parameters

**Table 25-7 SEND Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_to</td>
<td>Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list</td>
</tr>
<tr>
<td>p_from</td>
<td>Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent</td>
</tr>
<tr>
<td>p_body</td>
<td>Body of the email in plain text, not HTML (required). If a value is passed to <code>p_body_html</code>, then this is the only text the recipient sees. If a value is not passed to <code>p_body_html</code>, 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.</td>
</tr>
<tr>
<td>p_body_html</td>
<td>Body of the email in HTML format. This must be a full HTML document including the <code>&lt;html&gt;</code> and <code>&lt;body&gt;</code> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)</td>
</tr>
<tr>
<td>p_subj</td>
<td>Subject of the email</td>
</tr>
<tr>
<td>p_cc</td>
<td>Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list</td>
</tr>
</tbody>
</table>
Table 25-7  (Cont.) SEND Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_bcc</td>
<td>Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list</td>
</tr>
<tr>
<td>p_replyto</td>
<td>Address of the Reply-To mail header. You can use this parameter as follows:</td>
</tr>
<tr>
<td></td>
<td>• If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter</td>
</tr>
<tr>
<td></td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td>• If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you send these messages, but the automatic replies go to the value specified (for example, the email address)</td>
</tr>
</tbody>
</table>

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application and return the unique message ID.

```sql
-- Example One: Plain Text only message
DECLARE
  l_body      CLOB;
  l_id NUMBER;
BEGIN
  l_body := 'Thank you for your interest in the APEX_MAIL package.'||utl_tcp.crlf||utl_tcp.crlf;
  l_body := l_body || 'Sincerely,\n  The Application Express Dev Team'\n  l_id := apex_mail.send(
    p_to       => 'some_user@somewhere.com',   -- change to your email address
    p_from     => 'some_sender@somewhere.com', -- change to a real senders email address
    p_body     => l_body,
    p_subj     => 'APEX_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use APEX_MAIL.SEND to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses utl_tcp.crlf.

```sql
-- Example Two: Plain Text / HTML message
DECLARE
  l_body      CLOB;
  l_body_html CLOB;
  l_id NUMBER;
BEGIN
  l_body := 'To view the content of this message, please use an HTML
  Chapter 25
  SEND Function
  25-15'
enabled mail client.'||util_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;}
  </style>
  span.sig{font-style:italic;
      font-weight:bold;
      color:#811919;}
</head>
<body>
<p>Thank you for your interest in the <strong>APEX_MAIL</strong> package.</p>
Sincerely,<br />
<span class="sig">The Application Express Dev Team</span>
</body></html>||util_tcp.crlf;

l_id        := apex_mail.send(
  p_to        => 'some_user@somewhere.com',  -- change to your email address
  p_from      => 'some_sender@somewhere.com', -- change to a real senders email address
  p_body      => l_body,
  p_body_html => l_body_html,
  p_subj      => 'APEX_MAIL Package - HTML formatted message');
END;
/
APEX_PKG_APP_INSTALL

The APEX_PKG_APP_INSTALL package provides utilities you can use to manage Packaged Applications. You can use APEX_PKG_APP_INSTALL package to install, upgrade, and deinstall packaged application from the command line. Before using the package, the security_group_id must be set by calling either APEX_UTIL.SET_WORKSPACE or APEX_UTIL.SET_SECURITY_GROUP_ID. The related APEX views to get more information on packaged applications are APEX_PKG_APP_AUTHENTICATIONS, APEX_PKG_APP_AVAILABLE, APEX_PKG_APP_INSTALLED, and APEX_PKG_APP_ACTIVITY.

Note:
APEX_PKG_APP_INSTALL supports managing database packaged applications and does not support websheet packaged applications.

- DEINSTALL Procedure Signature 1
- DEINSTALL Procedure Signature 2
- INSTALL Function Signature 1
- INSTALL Function Signature 2
- UPGRADE Procedure Signature 1
- UPGRADE Procedure Signature 2

26.1 DEINSTALL Procedure Signature 1

This procedure deinstalls specified packaged application. If invalid packaged application ID passed, the procedure raises an error.

Syntax

APEX_PKG_APP_INSTALL.DEINSTALL(
    p_app_id in number);

Parameters

Table 26-1 DEINSTALL Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>The packaged application ID.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use DEINSTALL procedure to deinstall packaged application ID 7060 from DEMO workspace.

```
begin
    APEX_UTIL.SET_WORKSPACE( 'DEMO' );
    APEX_PKG_APP_INSTALL.DEINSTALL( p_app_id => 7060 );
end;
```

### 26.2 DEINSTALL Procedure Signature 2

This procedure deinstalls specified packaged application. If invalid packaged application ID passed, the procedure raises an error.

**Syntax**

```
APEX_PKG_APP_INSTALL.DEINSTALL( p_app_name in varchar2 );
```

**Parameters**

Table 26-2  DEINSTALL Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_name</td>
<td>The case insensitive packaged application name.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use DEINSTALL procedure to deinstall packaged application 'Bug Tracking' from DEMO workspace.

```
begin
    APEX_UTIL.SET_WORKSPACE( 'DEMO' );
    APEX_PKG_APP_INSTALL.DEINSTALL( p_app_name => 'Bug Tracking' );
end;
```

### 26.3 INSTALL Function Signature 1

This function installs specified packaged application and returns installed application ID. If any of the following are true, the function raises an error:

- Invalid packaged application ID passed.
- Invalid authentication type passed. For list of available authentication types, query APEX_PKG_APP_AUTHENTICATIONS view.
Syntax

APEX_PKG_APP_INSTALL.INSTALL(  
    p_app_id              in number,  
    p_authentication_type in  
    wwv_flow_authentication_api.t_authentication_type default null,  
    p_schema              in varchar2 default null ) return number;

Parameters

Table 26-3  INSTALL Function Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>The packaged application ID.</td>
</tr>
<tr>
<td>p_authentication_type</td>
<td>The type of authentication to use. If NULL, Oracle APEX builder authentication is used. The value must be apex_authentication.t_authentication_type.</td>
</tr>
<tr>
<td>p_schema</td>
<td>The database schema the application parses SQL statements as. If NULL, it defaults to the workspace schema. If there are more than one workspace schema, it defaults to the first schema provisioned for the workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use INSTALL function to install packaged application ID 7060 in DEMO workspace with 'Application Express Account' authentication using 'SCOTT' schema.

declare
    l_installed_app_id   number;
begin
    APEX_UTIL.SET_WORKSPACE( 'DEMO' );
    l_installed_app_id := APEX_PKG_APP_INSTALL.INSTALL(  
        p_app_id              => 7060,  
        p_authentication_type => APEX_AUTHENTICATION.C_TYPE_APEX_ACCOUNTS,  
        p_schema              => 'SCOTT'  
    );
end;

26.4 INSTALL Function Signature 2

This function installs specified packaged application and returns installed application ID. If any of the following are true, the function raises an error:

- Invalid packaged application ID passed.
- Invalid authentication type passed. For list of available authentication types, query APEX_PKG_APP_AUTHENTICATIONS view.
Syntax

APEX_PKG_APP_INSTALL.INSTALL(
    p_app_name            in varchar2,
    p_authentication_type in
    wwv_flow_authentication_api.t_authentication_type default null,
    p_schema              in varchar2 default null ) return number;

Parameters

Table 26-4 INSTALL Function Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_name</td>
<td>The case insensitive packaged application name.</td>
</tr>
<tr>
<td>p_authentication_type</td>
<td>The type of authentication to use. If NULL, Oracle APEX builder authentication is used. The value must be apex_authentication.t_authentication_type.</td>
</tr>
<tr>
<td>p_schema</td>
<td>The database schema the application parses SQL statements as. If NULL, it defaults to the workspace schema. If there are more than one workspace schema, it defaults to the first schema provisioned for the workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use INSTALL function to install packaged application named 'Bug Tracking' in DEMO workspace with 'Application Express Account' authentication using 'SCOTT' schema.

```
declare
    l_installed_app_id   number;
begin
    APEX_UTIL.SET_WORKSPACE( 'DEMO' );
    l_installed_app_id := APEX_PKG_APP_INSTALL.INSTALL(
        p_app_name => 'Bug Tracking',
        p_authentication_type => APEX_AUTHENTICATION.C_TYPE_APEXAccounts,
        p_schema => 'SCOTT' );
end;
```

26.5 UPGRADE Procedure Signature 1

This procedure upgrades specified packaged application. If any of the following are true, the procedure raises an error:

- Invalid packaged application ID passed.
- The installed packaged application is unlocked and cannot be upgraded.
- The installed packaged application is up to date and no upgrade needed.
Syntax

APEX_PKG_APP_INSTALL.UPGRADE(
    p_app_id in number);

Parameters

Table 26-5 UPGRADE Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>The packaged application ID.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use UPGRADE procedure to upgrade packaged application ID 7060 in DEMO workspace.

begin
    APEX_UTIL.SET_WORKSPACE( 'DEMO' );
    APEX_PKG_APP_INSTALL.UPGRADE(
        p_app_id => 7060 );
end;

26.6 UPGRADE Procedure Signature 2

This procedure upgrades specified packaged application. If any of the following are true, the procedure raises an error:

- Invalid packaged application ID passed.
- The installed packaged application is unlocked and cannot be upgraded.
- The installed packaged application is up to date and no upgrade needed.

Syntax

APEX_PKG_APP_INSTALL.UPGRADE(
    p_app_name in varchar2 );

Parameters

Table 26-6 UPGRADE Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_name</td>
<td>The case insensitive packaged application name.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use \texttt{UPGRADE} procedure to upgrade packaged application `Bug Tracking` in \texttt{DEMO} workspace.

\begin{verbatim}
begin
  APEX_UTIL.SET_WORKSPACE( 'DEMO' );
  APEX_PKG_APP_INSTALL.UPGRADE(
    p_app_name => 'Bug Tracking' );
end;
\end{verbatim}
The APEX_PAGE package is the public API for handling pages.

- Global Constants
- IS_DESKTOP_UI Function
- IS_JQM_SMARTPHONE_UI Function [DEPRECATED]
- IS_JQM_TABLET_UI Function [DEPRECATED]
- GET_UI_TYPE Function
- IS_READ_ONLY Function
- GET_PAGE_MODE Function
- PURGE_CACHE Procedure
- GET_URL Function

### 27.1 Global Constants

The following constants are used by this package.

```sql
c_ui_type_desktop        constant varchar2(10) := 'DESKTOP';
c_ui_type_jqm_smartphone constant varchar2(15) := 'JQM_SMARTPHONE';
```

### 27.2 IS_DESKTOP_UI Function

This function returns TRUE if the current page has been designed for desktop browsers.

**Syntax**

```sql
FUNCTION IS_DESKTOP_UI
RETURN BOOLEAN;
```

### 27.3 IS_JQM_SMARTPHONE_UI Function [DEPRECATED]

This function returns TRUE if the current page has been designed for smartphone devices using jQuery Mobile.

**Syntax**

```sql
FUNCTION IS_JQM_SMARTPHONE_UI
RETURN BOOLEAN;
```
27.4 IS_JQM_TABLET_UI Function [DEPRECATED]

This function returns TRUE if the current page has been designed for tablet devices using jQuery Mobile.

Syntax

FUNCTION IS_JQM_TABLET_UI
RETURN BOOLEAN;

27.5 GET_UI_TYPE Function

This function returns the user interface (UI) type for which the current page has been designed.

Syntax

FUNCTION GET_UI_TYPE
RETURN VARCHAR2;

27.6 IS_READ_ONLY Function

This function returns TRUE if the current page is rendered read-only and FALSE if it is not.

Syntax

FUNCTION IS_READ_ONLY
RETURN BOOLEAN;

27.7 GET_PAGE_MODE Function

This function returns the page mode for the current page.

Syntax

FUNCTION GET_PAGE_MODE (  
   p_application_id IN NUMBER,  
   p_page_id IN NUMBER)  
RETURN VARCHAR2;

Parameters

Table 27-1  GET_PAGE_MODE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>ID of the application. Defaults to the current application.</td>
</tr>
</tbody>
</table>
Table 27-1  (Cont.) GET_PAGE_MODE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>ID of the page. Defaults to the current page.</td>
</tr>
</tbody>
</table>

27.8 PURGE_CACHE Procedure

This procedure purges the cache of the specified application, page, and region for the specified user. If the user is not specified, the procedure purges all cached versions of the page.

Syntax

PROCEDURE PURGE_CACHE (
    p_application_id       IN NUMBER DEFAULT apex.g_flow_id,
    p_page_id              IN NUMBER DEFAULT apex.g_flow_step_id,
    p_user_name            IN VARCHAR2 DEFAULT NULL,
    p_current_session_only IN BOOLEAN  DEFAULT FALSE );

Parameters

Table 27-2  PURGE_CACHE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>ID of the application. Defaults to the current application.</td>
</tr>
<tr>
<td>p_page_id</td>
<td>ID of the page. Defaults to the current page. If you pass NULL, Oracle Application Express purges the cache on all pages of the application.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>Specify a user name if you only want to purge entries that were saved for the given user.</td>
</tr>
<tr>
<td>p_current_session_only</td>
<td>Specify TRUE if you only want to purge entries that were saved for the current session. Defaults to FALSE.</td>
</tr>
</tbody>
</table>

Example

This example purges session specific cache on the current page.

BEGIN
    APEX_PAGE.PURGE_CACHE (p_current_session_only => true );
END;

27.9 GET_URL Function

This function returns an Oracle Application Express f?p= URL. It is sometimes clearer to read a function call than a concatenated URL. See the example below for a comparison.
### Syntax

```sql
FUNCTION GET_URL (  
p_application        IN VARCHAR2 DEFAULT NULL,  
p_page               IN VARCHAR2 DEFAULT NULL,  
p_session            IN NUMBER   DEFAULT APEX.G_INSTANCE,  
p_request            IN VARCHAR2 DEFAULT NULL,  
p_debug              IN VARCHAR2 DEFAULT NULL,  
p_clear_cache        IN VARCHAR2 DEFAULT NULL,  
p_items              IN VARCHAR2 DEFAULT NULL,  
p_values             IN VARCHAR2 DEFAULT NULL,  
p_printer_friendly   IN VARCHAR2 DEFAULT NULL,  
p_trace              IN VARCHAR2 DEFAULT NULL )  
RETURN VARCHAR2;
```

### Parameters

**Table 27-3  GET_URL Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The application ID or alias. Defaults to the current application.</td>
</tr>
<tr>
<td>p_page</td>
<td>Page ID or alias. Defaults to the current page.</td>
</tr>
<tr>
<td>p_session</td>
<td>Session ID. Defaults to the current session ID.</td>
</tr>
<tr>
<td>p_request</td>
<td>URL request parameter.</td>
</tr>
<tr>
<td>p_debug</td>
<td>URL debug parameter. Defaults to the current debug mode.</td>
</tr>
<tr>
<td>p_clear_cache</td>
<td>URL clear cache parameter.</td>
</tr>
<tr>
<td>p_items</td>
<td>Comma-delimited list of item names to set session state.</td>
</tr>
<tr>
<td>p_values</td>
<td>Comma-delimited list of item values to set session state.</td>
</tr>
<tr>
<td>p_printer_friendly</td>
<td>URL printer friendly parameter. Defaults to the current request's printer friendly mode.</td>
</tr>
<tr>
<td>p_trace</td>
<td>SQL trace parameter.</td>
</tr>
</tbody>
</table>

### Example

This query uses APEX_PAGE.GET_URL and its alternative APEX_UTIL.PREPARE_URL to produce two identical URLs.

```sql
SELECT APEX_PAGE.GET_URL (  
    p_page => 1,  
    p_items => 'P1_X,P1_Y',  
    p_values => 'somevalue,othervalue' ) f_url_1,  
APEX_UTIL.PREPARE_URL(  
    'f?p=&APP_ID.:  
1:&APP_SESSION.:::P1_X,P1_Y:somevalue,othervalue')
FROM DUAL
```
The APEX_PLUGIN package provides the interface declarations and some utility functions to work with plug-ins.

- Data Types
- GET_AJAX_IDENTIFIER Function
- GET_INPUT_NAME_FOR_PAGE_ITEM Function

### 28.1 Data Types

The data types used by the APEX_PLUGIN package are described in this section.

**Data Types:**

- `c_*`
- `t_authentication`
- `t_authentication_ajax_result`
- `t_authentication_auth_result`
- `t_authentication_inval_result`
- `t_authentication_logout_result`
- `t_authentication_sentry_result`
- `t_authorization`
- `t_authorization_exec_result`
- `t_dynamic_action`
- `t_dynamic_action_ajax_result`
- `t_dynamic_action_render_result`
- `t_page_item`
- `t_page_item_ajax_result`
- `t_page_item_render_result`
- `t_page_item_validation_result`
- `t_plugin`
- `t_process`
- `t_process_exec_result`
- `type t_region_column (`
  `type t_region_columns is table of t_region_column index by pls_integer;`
  `type t_region_column (`
The following constants are used for display_location in the page item validation function result type t_page_item_validation_result.

```plaintext
c_inline_with_field constant varchar2(40) := 'INLINE_WITH_FIELD';
c_inline_with_field_and_notif constant varchar2(40) := 'INLINE_WITH_FIELD_AND_NOTIFICATION';
c_inline_in_notification constant varchar2(40) := 'INLINE_IN_NOTIFICATION';
c_on_error_page constant varchar2(40) := 'ON_ERROR_PAGE';
```

### t_authentication

```plaintext
type t_authentication is record (    id                   number,    name                 varchar2(255),    invalid_session_url  varchar2(4000),    logout_url           varchar2(4000),    plsql_code           clob,    attribute_01         varchar2(32767),    attribute_02         varchar2(32767),    attribute_03         varchar2(32767),    attribute_04         varchar2(32767),    attribute_05         varchar2(32767),    attribute_06         varchar2(32767),    attribute_07         varchar2(32767),    attribute_08         varchar2(32767),    attribute_09         varchar2(32767),    attribute_10         varchar2(32767),    attribute_11         varchar2(32767),    attribute_12         varchar2(32767),    attribute_13         varchar2(32767),    attribute_14         varchar2(32767),    attribute_15         varchar2(32767),    --    session_id           number,    username             varchar2(255) );
```

### t_authentication_ajax_result

```plaintext
type t_authentication_ajax_result is record (    dummy                boolean );
```

### t_authentication_auth_result

```plaintext
type t_authentication_auth_result is record (    is_authenticated     boolean,    redirect_url         varchar2(4000),    log_code             number,    log_text             varchar2(4000),    display_text         varchar2(4000) );
```
**t_authentication_inval_result**

type t_authentication_inval_result is record (  
  redirect_url varchar2(4000) );

**t_authentication_logout_result**

type t_authentication_logout_result is record (  
  redirect_url varchar2(4000) );

**t_authentication_sentry_result**

type t_authentication_sentry_result is record (  
  is_valid boolean );

**t_authorization**

The following type is passed to all authorization plug-in functions and contains information about the current authorization.

type t_authorization is record (  
  id number,  
  name varchar2(255),  
  username varchar2(255),  
  caching varchar2(20),  
  component apex.t_component,  
  attribute_01 varchar2(32767),  
  attribute_02 varchar2(32767),  
  attribute_03 varchar2(32767),  
  attribute_04 varchar2(32767),  
  attribute_05 varchar2(32767),  
  attribute_06 varchar2(32767),  
  attribute_07 varchar2(32767),  
  attribute_08 varchar2(32767),  
  attribute_09 varchar2(32767),  
  attribute_10 varchar2(32767),  
  attribute_11 varchar2(32767),  
  attribute_12 varchar2(32767),  
  attribute_13 varchar2(32767),  
  attribute_14 varchar2(32767),  
  attribute_15 varchar2(32767),

**t_authorization_exec_result**

The `t_authorization_exec_result` data type has been added to the APEX_PLUGIN package.

type t_authorization_exec_result is record (  
  is_authorized boolean );

**t_dynamic_action**

The following type is passed into all dynamic action plug-in functions and contains information about the current dynamic action.

type t_dynamic_action is record (  
  id number,  
  action varchar2(50),  
  attribute_01 varchar2(32767),  
  attribute_02 varchar2(32767),
The following type is used as the result type for the Ajax function of a dynamic action plug-in.

type t_dynamic_action_ajax_result is record (  
dummy boolean /* not used yet */  );

tDynamic_action_render_result

The following type is used as the result type for the rendering function of a dynamic action plug-in.

type t_dynamic_action_render_result is record (  
javascript_function varchar2(32767),  
ajax_identifier varchar2(255),  
attribute_01 varchar2(32767),  
attribute_02 varchar2(32767),  
attribute_03 varchar2(32767),  
attribute_04 varchar2(32767),  
attribute_05 varchar2(32767),  
attribute_06 varchar2(32767),  
attribute_07 varchar2(32767),  
attribute_08 varchar2(32767),  
attribute_09 varchar2(32767),  
attribute_10 varchar2(32767),  
attribute_11 varchar2(32767),  
attribute_12 varchar2(32767),  
attribute_13 varchar2(32767),  
attribute_14 varchar2(32767),  
attribute_15 varchar2(32767) );

t_page_item

The following type is passed into all item type plug-in functions and contains information about the current page item.

type t_page_item is record (  
id                          number,  
name                        varchar2(255),  
label                       varchar2(4000),  
plain_label                 varchar2(4000),  
label_id                    varchar2(255), /* label id is set if ©Standard Form Element© = no and label template uses #LABEL_ID# substitution */  
placeholder                 varchar2(255),  
format_mask                 varchar2(255),  
);
is_required boolean,
lov_definition varchar2(4000),
lov_display_extra boolean,
lov_display_null boolean,
lov_null_text varchar2(255),
lov_null_value varchar2(255),
lov_cascade_parent_items varchar2(255),
aajax_items_to_submit varchar2(255),
aajax_optimize_refresh boolean,
element_width number,
element_max_length number,
element_height number,
element_css_classes varchar2(255),
element_attributes varchar2(2000),
element_option_attributes varchar2(4000),
escape_output boolean,
attribute_01 varchar2(32767),
attribute_02 varchar2(32767),
attribute_03 varchar2(32767),
attribute_04 varchar2(32767),
attribute_05 varchar2(32767),
attribute_06 varchar2(32767),
attribute_07 varchar2(32767),
attribute_08 varchar2(32767),
attribute_09 varchar2(32767),
attribute_10 varchar2(32767),
attribute_11 varchar2(32767),
attribute_12 varchar2(32767),
attribute_13 varchar2(32767),
attribute_14 varchar2(32767),
attribute_15 varchar2(32767) );

t_page_item_ajax_result

The following type is used as the result type for the Ajax function of an item type plug-in.

type t_page_item_ajax_result is record (
    dummy boolean /* not used yet */
);
page_item_name varchar2(255) ); /* if not set the validated page item name is used */

**t_plugin**
The following type is passed into all plug-in functions and contains information about the current plug-in.

```plaintext
t_plugin is record (  
  name         varchar2(45),  
  file_prefix  varchar2(4000),  
  attribute_01 varchar2(32767),  
  attribute_02 varchar2(32767),  
  attribute_03 varchar2(32767),  
  attribute_04 varchar2(32767),  
  attribute_05 varchar2(32767),  
  attribute_06 varchar2(32767),  
  attribute_07 varchar2(32767),  
  attribute_08 varchar2(32767),  
  attribute_09 varchar2(32767),  
  attribute_10 varchar2(32767),  
  attribute_11 varchar2(32767),  
  attribute_12 varchar2(32767),  
  attribute_13 varchar2(32767),  
  attribute_14 varchar2(32767),  
  attribute_15 varchar2(32767) );
```

**t_process**
The following type is passed into all process type plug-in functions and contains information about the current process.

```plaintext
t_process is record (  
  id                   number,  
  name                 varchar2(255),  
  success_message      varchar2(32767),  
  attribute_01         varchar2(32767),  
  attribute_02         varchar2(32767),  
  attribute_03         varchar2(32767),  
  attribute_04         varchar2(32767),  
  attribute_05         varchar2(32767),  
  attribute_06         varchar2(32767),  
  attribute_07         varchar2(32767),  
  attribute_08         varchar2(32767),  
  attribute_09         varchar2(32767),  
  attribute_10         varchar2(32767),  
  attribute_11         varchar2(32767),  
  attribute_12         varchar2(32767),  
  attribute_13         varchar2(32767),  
  attribute_14         varchar2(32767),  
  attribute_15         varchar2(32767) );
```

**t_process_exec_result**
The following type is used as the result type for the execution function of a process type plug-in.

```plaintext
t_process_exec_result is record (  
  success_message varchar2(32767),  
  execution_skipped boolean default false /* set to TRUE if process execution has */
```
been skipped by plug-in because of additional condition checks */

); type t_region_column ( The following type is passed into all region type plug-in functions and contains information about the current region.

type t_region_column is record ( id                   number,
  name                 t_region_column_name,
  is_displayed         boolean,
  heading              apex_region_columns.heading%type,
  heading_alignment    apex_region_columns.heading_alignment%type,
  value_alignment      apex_region_columns.value_alignment%type,
  value_css_classes    apex_region_columns.value_css_classes%type,
  value_attributes     apex_region_columns.value_attributes%type,
  format_mask          apex_region_columns.format_mask%type,
  escape_output        boolean,
  attribute_01         varchar2(32767),
  attribute_02         varchar2(32767),
  attribute_03         varchar2(32767),
  attribute_04         varchar2(32767),
  attribute_05         varchar2(32767),
  attribute_06         varchar2(32767),
  attribute_07         varchar2(32767),
  attribute_08         varchar2(32767),
  attribute_09         varchar2(32767),
  attribute_10         varchar2(32767),
  attribute_11         varchar2(32767),
  attribute_12         varchar2(32767),
  attribute_13         varchar2(32767),
  attribute_14         varchar2(32767),
  attribute_15         varchar2(32767),
  attribute_16         varchar2(32767),
  attribute_17         varchar2(32767),
  attribute_18         varchar2(32767),
  attribute_19         varchar2(32767),
  attribute_20         varchar2(32767),
  attribute_21         varchar2(32767),
  attribute_22         varchar2(32767),
  attribute_23         varchar2(32767),
  attribute_24         varchar2(32767),
  attribute_25         varchar2(32767);

type t_region_columns is table of t_region_column index by pls_integer;

t_region ( The following type is passed into all region type plug-in functions and contains information about the current region.

type t_region is record ( id                   number,
  static_id             varchar2(255),
  name                  varchar2(255),
  type                  varchar2(255),
  source                varchar2(32767),
  ajax_items_to_submit  varchar2(32767),
  fetched_rows          pls_integer,
  fetched_rows         pls_integer,
The following type is used as result type for the Ajax function of a region type plug-in.

```plsql
type t_region_ajax_result is record (
    dummy boolean /* not used yet */
);
```

The following type is used as the result type for the rendering function of a region type plug-in.

```plsql
type t_region_render_result is record (
    navigable_dom_id varchar2(255) /* can be used to put focus to an input field (that is, search field) the region renders as part of the plug-in output */
);
```

## 28.2 GET_AJAX_IDENTIFIER Function

This function returns the Ajax identifier used to call the Ajax callback function defined for the plug-in.
Note:
This function only works in the context of a plug-in rendering function call and only if the plug-in has defined an Ajax function callback in the plug-in definition.

Syntax

APEX_PLUGIN.GET_AJAX_IDENTIFIER
RETURN VARCHAR2;

Parameters
None.

Example
This is an example of a dynamic action plug-in rendering function that supports an Ajax callback.

function render_set_value ( p_dynamic_action in apex_plugin.t_dynamic_action )
return apex_plugin.t_dynamic_action_render_result
is
l_result apex_plugin.t_dynamic_action_render_result;
begin
l_result.javascript_function := 'com_oracle_apex_set_value';
l_result.ajax_identifier := apex_plugin.get_ajax_identifier;
return l_result;
end;

28.3 GET_INPUT_NAME_FOR_PAGE_ITEM Function

Use this function when you want to render an HTML input element in the rendering function of an item type plug-in. For the HTML input element, for example, `<input type="text" id="P1_TEST" name="xxx">`, you have to provide a value for the name attribute so that Oracle Application Express can map the submitted value to the actual page item in session state. This function returns the mapping name for your page item. If the HTML input element has multiple values, such as a select list with `multiple="multiple"`, then set `p_is_multi_value` to TRUE.

Note:
This function is only useful when called in the rendering function of an item type plug-in.
Syntax

APEX_PLUGIN.GET_INPUT_NAME_FOR_PAGE_ITEM (  
    p_is_multi_value  IN BOOLEAN)  
RETURN VARCHAR2;

Parameters

Table 28-1  GET_INPUT_NAME_FOR_PAGE_ITEM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_is_multi_value</td>
<td>Set to TRUE if the HTML input element has multiple values. If not, set to FALSE. HTML input elements with multiple values can be checkboxes and multi select lists.</td>
</tr>
</tbody>
</table>

Example

The following example outputs the necessary HTML code to render a text field where the value gets stored in session state when the page is submitted.

```sql
sys.htp.prn (  
    '<input type="text" id="'||p_item.name||'" value="'||sys.htf.escape_sc(p_value)||'" size="'||p_item.element_width||'" maxlength="'||p_item.element_max_length||'" name="'||apex_plugin.get_input_name_for_page_item(false)||'" class="text_field"'/>');
```
APEX_PLUGIN_UTIL

The APEX_PLUGIN_UTIL package provides utility functions that solve common problems when writing a plug-in.

- CLEAR_COMPONENT_VALUES Procedure
- DEBUG_DYNAMIC_ACTION Procedure
- DEBUG_PAGE_ITEM Procedure Signature 1
- DEBUG_PAGE_ITEM Procedure Signature 2
- DEBUG_PROCESS Procedure
- DEBUG_REGION Procedure Signature 1
- DEBUG_REGION Procedure Signature 2
- ESCAPE Function
- EXECUTE_PLSQL_CODE Procedure
- GET_ATTRIBUTE_AS_NUMBER Function
- GET_DATA Function Signature 1
- GET_DATA Function Signature 2
- GET_DATA2 Function Signature 1
- GET_DATA2 Function Signature 2
- GET_DISPLAY_DATA Function Signature 1
- GET_DISPLAY_DATA Function Signature 2
- GET_ELEMENT_ATTRIBUTES Function
- GET_PLSQL_EXPRESSION_RESULT Function
- GET_PLSQL_FUNCTION_RESULT Function
- GET_POSITION_IN_LIST Function
- GET_SEARCH_STRING Function
- GET_VALUE_AS_VARCHAR2 Function
- IS_EQUAL Function
- PAGE_ITEM_NAMES_TO_JQUERY Function
- PRINT_DISPLAY_ONLY Procedure
- PRINT_ESCAPED_VALUE Procedure
- PRINT_HIDDEN_IF_READONLY Procedure
- PRINT_JSON_HTTP_HEADER Procedure
- PRINT_LOV_AS_JSON Procedure
- PRINT_OPTION Procedure
29.1 CLEAR_COMPONENT_VALUES Procedure

This procedure clears the component specific Session State set by `apex_plugin_util.set_component_values`.

**Syntax**

```plaintext
procedure clear_component_values;
```

**Example**

See `apex_plugin_util.set_component_values`

---

29.2 DEBUG_DYNAMIC_ACTION Procedure

This procedure writes the data of the dynamic action meta data to the debug output if debugging is enabled.

**Syntax**

```plaintext
APEX_PLUGIN_UTIL.DEBUG_DYNAMIC_ACTION ( 
  p_plugin         IN apex_plugin.t_plugin,
  p_dynamic_action IN apex_plugin.t_dynamic_action);
```

**Parameters**

**Table 29-1  DEBUG_DYNAMIC_ACTION Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plugin</td>
<td>This is the p_plugin parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_dynamic_action</td>
<td>This is the p_dynamic_action parameter of your plug-in function.</td>
</tr>
</tbody>
</table>
Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the rendered function or Ajax callback function of the plug-in.

```plaintext
apex_plugin_util.debug_dynamic_action (  
    p_plugin         => p_plugin,  
    p_dynamic_action => p_dynamic_action );
```

### 29.3 DEBUG_PAGE_ITEM Procedure Signature 1

This procedure writes the data of the page item meta data to the debug output if debugging is enabled.

#### Syntax

```plaintext
APEX_PLUGIN_UTIL.DEBUG_PAGE_ITEM (  
    p_plugin    IN apex_plugin.t_plugin,  
    p_page_item IN apex_plugin.t_page_item);  
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plugin</td>
<td>This is the <code>p_plugin</code> parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_page_item</td>
<td>This is the <code>p_page_item</code> parameter of your plug-in function.</td>
</tr>
</tbody>
</table>

**Example**

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the renderer, Ajax callback or validation function.

```plaintext
apex_plugin_util.debug_page_item (  
    p_plugin    => p_plugin,  
    p_page_item => p_page_item );
```

### 29.4 DEBUG_PAGE_ITEM Procedure Signature 2

This procedure writes the data of the page item meta data to the debug output if debugging is enabled.

#### Syntax

```plaintext
APEX_PLUGIN_UTIL.DEBUG_PAGE_ITEM (  
    p_plugin              IN apex_plugin.t_plugin,  
    p_page_item           IN apex_plugin.t_page_item,  
```
p_value               IN VARCHAR2,
p_is_readonly         IN BOOLEAN,
p_is_printer_friendly IN BOOLEAN);

Parameters

Table 29-3  DEBUG_PAGE_ITEM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plugin</td>
<td>This is the p_plugin parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_page_item</td>
<td>This is the p_page_item parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_value</td>
<td>This is the p_value parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_is_readonly</td>
<td>This is the p_is_readonly parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_is_printer_friendly</td>
<td>This is the p_is_printer_friendly parameter of your plug-in function.</td>
</tr>
</tbody>
</table>

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the renderer, Ajax callback or validation function.

apex_plugin_util.debug_page_item (  
p_plugin => p_plugin,  
p_page_item => p_page_item,  
p_value => p_value,  
p_is_readonly => p_is_readonly,  
p_is_printer_friendly => p_is_printer_friendly);

29.5 DEBUG_PROCESS Procedure

This procedure writes the data of the process meta data to the debug output if debugging is enabled.

Syntax

APEX_PLUGIN_UTIL.DEBUG_PROCESS (  
p_plugin IN apex_plugin.t_plugin,  
p_process IN apex_plugin.t_process);

Parameters

Table 29-4  DEBUG_PROCESS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plugin</td>
<td>This is the p_plugin parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_process</td>
<td>This is the p_process parameter of your plug-in function.</td>
</tr>
</tbody>
</table>
Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the execution function of the plug-in.

```apl
apex_plugin_util.debug_process (    p_plugin => p_plugin,    p_process => p_process);
```

29.6 DEBUG_REGION Procedure Signature 1

This procedure writes the data of the region meta data to the debug output if debugging is enabled.

Syntax

```apl
APEX_PLUGIN_UTIL.DEBUG_REGION (    p_plugin              IN apex_plugin.t_plugin,    p_region              IN apex_plugin.t_region);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plugin</td>
<td>This is the p_plugin parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_region</td>
<td>This is the p_region parameter of your plug-in function.</td>
</tr>
</tbody>
</table>

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the render function or Ajax callback function of the plug-in.

```apl
apex_plugin_util.debug_process (    p_plugin => p_plugin,    p_region => p_region);
```

29.7 DEBUG_REGION Procedure Signature 2

This procedure writes the data of the region meta data to the debug output if debugging is enabled. This is the advanced version of the debugging procedure which is used for the rendering function of a region plug-in.

Syntax

```apl
APEX_PLUGIN_UTIL.DEBUG_REGION (    p_plugin                  IN apex_plugin.t_plugin,
    p_region                  IN apex_plugin.t_region,
```
p_region              IN apex_plugin.t_region,
p_is_printer_friendly IN BOOLEAN);

Parameters

Table 29-6 describes the parameters available in the DEBUG_REGION procedure.

Table 29-6  DEBUG_REGION Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plugin</td>
<td>This is the p_plugin parameter of your plug-in function</td>
</tr>
<tr>
<td>p_region</td>
<td>This is the p_region parameter of your plug-in function</td>
</tr>
<tr>
<td>p_is_printer_friendly</td>
<td>This is the p_is_printer_friendly parameter of your plug-in function</td>
</tr>
</tbody>
</table>

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the render function or Ajax callback function of the plug-in.

apex_plugin_util.debug_region (
    p_plugin => p_plugin,
    p_region => p_region,
    p_is_printer_friendly => p_is_printer_friendly);

29.8 ESCAPE Function

This function is used if you have checked the standard attribute "Has Escape Output Attribute" option for your item type plug-in which allows a developer to decide if the output should be escaped or not.

Syntax

APEX_PLUGIN_UTIL.ESCAPE (  
    p_value  IN VARCHAR2,
    p_escape IN BOOLEAN)  
RETURN VARCHAR2;

Parameters

Table 29-7  ESCAPE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>This is the value you want to escape depending on the p_escape parameter.</td>
</tr>
<tr>
<td>p_escape</td>
<td>If set to TRUE, the return value is escaped. If set to FALSE, the value is not escaped.</td>
</tr>
</tbody>
</table>
Example

This example outputs all values of the array `l_display_value_list` as a HTML list and escapes the value of the array depending on the setting the developer as picked when using the plug-in.

```plsql
for i in 1 .. l_display_value_list.count
  loop
    sys.htp.prn ('<li>',
      apex_plugin_util.escape (
        p_value  => l_display_value_list(i),
        p_escape => p_item.escape_output ),
      '</li>')
  end loop;
```

29.9 EXECUTE_PLSQL_CODE Procedure

This procedure executes a PL/SQL code block and performs binding of bind variables in the provided PL/SQL code. This procedure is usually used for plug-in attributes of type PL/SQL Code.

Syntax

```plsql
APEX_PLUGIN_UTIL.EXECUTE_PLSQL_CODE ( p_plsql_code  IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plsql_code</td>
<td>PL/SQL code to be executed.</td>
</tr>
</tbody>
</table>

Example

Text which should be escaped and then printed to the HTTP buffer.

```plsql
declare
  l_plsql_code VARCHAR(32767) := p_process.attribute_01;
begin
  apex_plugin_util.execute_plsql_code ( p_plsql_code => l_plsql_code );
end;
```

29.10 GET_ATTRIBUTE_AS_NUMBER Function

This function returns the value of a plug-in attribute as a number, taking into account NLS decimal separator effective for the current database session. Use this function in
plug-in PL/SQL source for custom attributes of type NUMBER instead of the built-in to_number function.

Syntax

APEX_PLUGIN_UTIL.GET_ATTRIBUTE_AS_NUMBER (
   p_value IN VARCHAR2,
   p_attribute_label IN VARCHAR2 )
return NUMBER;

Parameters

Table 29-9  GET_ATTRIBUTE_AS_NUMBER Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_attribute_label</td>
<td>The label of the custom plug-in attribute.</td>
</tr>
<tr>
<td>p_value</td>
<td>The value of a custom attribute of type NUMBER.</td>
</tr>
</tbody>
</table>

Example

```sql
declare
    l_value number;
begin
    -- The following may fail for languages that don't use dot as the NLS
decimal separator
    l_value := to_number( p_region.attribute_04 );

    -- The following will work correctly regardless of the effective NLS
decimal separator
    l_value := apex_plugin_util.get_attribute_as_number( p_region.attribute_04, 'Minimum Amount' );
end;
/
```

29.11 GET_DATA Function Signature 1

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned as a string, independent of their data types. The search column is identified by providing a column number in the p_search_column_no parameter. This function takes into account character value comparison globalization attributes defined for the application.

Syntax

APEX_PLUGIN_UTIL.GET_DATA (
   p_sql_statement    IN VARCHAR2,
   p_min_columns      IN NUMBER,
   p_max_columns      IN NUMBER,
   p_component_name   IN VARCHAR2,
   p_search_type      IN VARCHAR2 DEFAULT 2,
p_search_column_no IN VARCHAR2 DEFAULT 2,
p_search_string    IN VARCHAR2 DEFAULT NULL,
p_first_row        IN NUMBER DEFAULT NULL,
p_max_rows         IN NUMBER DEFAULT NULL)
RETURN t_column_value_list;

Parameters

Table 29-10  GET_DATA Function Signature 1Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>SQL statement used for the lookup.</td>
</tr>
<tr>
<td>p_min_columns</td>
<td>Minimum number of return columns.</td>
</tr>
<tr>
<td>p_max_columns</td>
<td>Maximum number of return columns.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>In case an error is returned, this is the name of the page item or report</td>
</tr>
<tr>
<td></td>
<td>column used to display the error message.</td>
</tr>
<tr>
<td>p_search_type</td>
<td>Must be one of the c_search_* constants. They are as follows:</td>
</tr>
<tr>
<td></td>
<td>c_search_contains_case, c_search_contains_ignore,</td>
</tr>
<tr>
<td></td>
<td>c_search_exact_case, c_search_exact_ignore</td>
</tr>
<tr>
<td>p_search_column_no</td>
<td>Number of the column used to restrict the SQL statement. Must be within the</td>
</tr>
<tr>
<td></td>
<td>p_min_columns though p_max_columns range.</td>
</tr>
<tr>
<td>p_search_string</td>
<td>Value used to restrict the query.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>Start query at the specified row. All rows before the specified row are</td>
</tr>
<tr>
<td></td>
<td>skipped.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum number of return rows allowed.</td>
</tr>
</tbody>
</table>

Return

Table 29-11  GET_DATA Function Signature 1 Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_column_value_list</td>
<td>Table of apex_application_global.vc_arr2 indexed by column number.</td>
</tr>
</tbody>
</table>

Example

The following example shows a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list.

```sql
function render_list (  
  p_item                in apex_plugin.t_page_item,
  p_value               in varchar2,
  p_is_readonly         in boolean,
  p_is_printer_friendly in boolean )  
return apex_plugin.t_page_item_render_result  
is  
t_column_value_list   apex_plugin_util.t_column_value_list;
```
begin
    l_column_value_list :=
        apex_plugin_util.get_data (  
            p_sql_statement => p_item.lov_definition,
            p_min_columns => 2,
            p_max_columns => 2,
            p_component_name => p_item.name,
            p_search_type => apex_plugin_util.c_search_contains_case,
            p_search_column_no => 1,
            p_search_string => p_value );

    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list(1).count loop
        sys.htp.p (  
            '<li>'||
            sys.htf.escape_sc(l_column_value_list(1)(i))|| -- display
            column
            '-'||
            sys.htf.escape_sc(l_column_value_list(2)(i))|| -- return column
            '</li>' );
    end loop;
    sys.htp.p('</ul>');
    end render_list;

29.12 GET_DATA Function Signature 2

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned as a string, independent of their data types. The search column is identified by providing a column name in the p_search_column_name parameter. This function takes into account character value comparison globalization attributes defined for the application.

Syntax

APEX_PLUGIN_UTIL.GET_DATA (  
    p_sql_statement IN VARCHAR2,
    p_min_columns IN NUMBER,
    p_max_columns IN NUMBER,
    p_component_name IN VARCHAR2,
    p_search_type IN VARCHAR2 DEFAULT NULL,
    p_search_column_name IN VARCHAR2 DEFAULT NULL,
    p_search_string IN VARCHAR2 DEFAULT NULL,
    p_first_row IN VARCHAR2 DEFAULT NULL,
    p_max_rows IN NUMBER DEFAULT NULL)  
RETURN t_column_value_list;
Parameters

Table 29-12  GET_DATA Function Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>SQL statement used for the lookup.</td>
</tr>
<tr>
<td>p_min_columns</td>
<td>Minimum number of return columns.</td>
</tr>
<tr>
<td>p_max_columns</td>
<td>Maximum number of return columns.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>In case an error is returned, this is the name of the page item or report column used to display the error message.</td>
</tr>
<tr>
<td>p_search_type</td>
<td>Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore</td>
</tr>
<tr>
<td>p_search_column_name</td>
<td>This is the column name used to restrict the SQL statement.</td>
</tr>
<tr>
<td>p_search_string</td>
<td>Value used to restrict the query.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>Start query at the specified row. All rows before the specified row are skipped.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum number of return rows allowed.</td>
</tr>
</tbody>
</table>

Return

Table 29-13  GET_TABLE Function Signature 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_column_value_list</td>
<td>Table of apex_application_global.vc_arr2 indexed by column number.</td>
</tr>
</tbody>
</table>

Example

The following example shows a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list.

```sql
function render_list ( p_item                in apex_plugin.t_page_item,
                      p_value               in varchar2,
                      p_is_readonly         in boolean,
                      p_is_printer_friendly in boolean )
return apex_plugin.t_page_item_render_result
is
  l_column_value_list   apex_plugin_util.t_column_value_list;
begin
  l_column_value_list :=
    apex_plugin_util.get_data ( p_sql_statement    => p_item.lov_definition,
                               p_min_columns      => 2,
                               p_max_columns      => 2,
                               p_component_name   => p_item.name,
                               p_search_type      => apex_plugin_util.c_search_contains_case,
                               p_search_column_name => null,
                               p_search_string    => p_value,
                               p_first_row        => 1,
                               p_max_rows         => 10 );
```
```java
    p_search_column_name => 'ENAME',
p_search_string => p_value );

    sys.htp.p('<ul>');</
for i in 1 .. l_column_value_list(1).count
  loop
    sys.htp.p(
      '<li>'||
      sys.htf.escape_sc(l_column_value_list(1)(i))|| -- display
column
      ' - '||
      sys.htf.escape_sc(l_column_value_list(2)(i))|| -- return column
      '</li>');
  end loop;
  sys.htp.p('</ul>');
end render_list;

29.13 GET_DATA2 Function Signature 1

Executes the specified SQL query restricted by the provided search string (optional)
and returns the values for each column. All column values are returned along with their
original data types. The search column is identified by providing a column number in
the p_search_column_no parameter. This function takes into account character value
comparison globalization attributes defines for the application.

Syntax

APEX_PLUGIN_UTIL.GET_DATA2 (  
p_sql_statement IN VARCHAR2,
p_min_columns IN NUMBER,
p_max_columns IN NUMBER,
p_data_type_list IN WWV_GLOBAL.VC_ARR2 DEFAULT C_EMPTY_DATA_TYPE_LIST,
p_component_name IN VARCHAR2,
p_search_type IN VARCHAR2 DEFAULT 2,
p_search_column_no IN VARCHAR2 DEFAULT 2,
p_search_string IN VARCHAR2 DEFAULT NULL,
p_first_row IN NUMBER DEFAULT NULL,
p_max_rows IN NUMBER DEFAULT NULL)
RETURN t_column_value_list2;

Parameters

Table 29-14  GET_DATA2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>SQL statement used for the lookup.</td>
</tr>
<tr>
<td>p_min_columns</td>
<td>Minimum number of return columns.</td>
</tr>
<tr>
<td>p_max_columns</td>
<td>Maximum number of return columns.</td>
</tr>
</tbody>
</table>
Table 29-14  (Cont.) GET_DATA2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_data_type_list</td>
<td>If provided, checks to make sure the data type for each column matches the specified data type in the array. Use the constants c_data_type_* for available data types.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>In case an error is returned, this is the name of the page item or report column used to display the error message.</td>
</tr>
<tr>
<td>p_search_type</td>
<td>Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore</td>
</tr>
<tr>
<td>p_search_column_no</td>
<td>Number of the column used to restrict the SQL statement. Must be within the p_min_columns though p_max_columns range.</td>
</tr>
<tr>
<td>p_search_string</td>
<td>Value used to restrict the query.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>Start query at the specified row. All rows before the specified row are skipped.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum number of return rows allowed.</td>
</tr>
</tbody>
</table>

Return

Table 29-15  GET_DATA2 Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_column_value_list2</td>
<td>Table of t_column_values indexed by column number.</td>
</tr>
</tbody>
</table>

Example

The following example is a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list. This time, the first column of the LOV SQL statement is checked if it is of type VARCHAR2 and the second is of type number.

```sql
function render_list ( p_item                in apex_plugin.t_page_item,
                       p_value               in varchar2,
                       p_is_readonly         in boolean,
                       p_is_printer_friendly in boolean )
return apex_plugin.t_page_item_render_result
is
  l_data_type_list    apex_application_global.vc_arr2;
  l_column_value_list apex_plugin_util.t_column_value_list2;
begin
  -- The first LOV column has to be a string and the second a number
  l_data_type_list(1) := apex_plugin_util.c_data_type_varchar2;
  l_data_type_list(2) := apex_plugin_util.c_data_type_number;

  --
  l_column_value_list :=
                    apex_plugin_util.get_data2 ( p_sql_statement => p_item.lov_definition,
                                          p_data_type_list => l_data_type_list,
                                          p_first_row      => 1, p_max_rows => 100,
                                          p_search_string  => p_value,
                                          p_search_type    => c_search_contains_case, p_search_column_no => 1, p_min_columns => 1,
                                          p_component_name => p_item.component_name,
                                          p_min_columns    => 1, p_max_columns => 100 );
end;
```

29-13
function render (  
    p_region in apex_plugin.t_region,  
    p_plugin in apex_plugin.t_plugin,  
    p_is_printer_friendly in boolean )  
return apex_plugin.t_region_render_result  
is  
    l_column_value_list apex_plugin_util.t_column_value_list2;  
    l_geometry sdo_geometry;  
    l_value varchar2(32767);  
    l_dummy pls_integer;  
begin  
    l_column_value_list :=  
        apex_plugin_util.get_data2 (  
            p_sql_statement => p_region.source,  
            p_min_columns => 1,  
            p_max_columns => null,  
            p_component_name => p_region.name );  
    --  
    sys.htp.p('<ul>');  
    for row in 1 .. l_column_value_list.count loop  
        sys.htp.p('<li>');  
        for col in 1 .. l_column_value_list.count loop  
            sys.htp.p(  
                sdo_geometry.value_list(col).varchar2_value ||  
                ' - ' ||  
                sdo_geometry.value_list(col).number_value ||  
                ' </li>');  
        end loop;  
        sys.htp.p('</ul>');  
    end loop;  
end render_list;

The following example is a simple region type plug-in rendering function which executes the SQL query defined for the region. The result is then generated as a HTML list. This example demonstrates the advanced handling of object type columns like SDO_GEOMETRY.
if l_column_value_list(col).data_type = 'SDO_GEOMETRY' then
    -- Object Type columns are always returned using ANYDATA
    and we have to
    -- use GETOBJECT to transform them back into the original
    object type
    l_dummy := l_column_value_list(col).value_list(row).anydata_value.getobject(l_geometry);
    l_value := '(' || l_geometry.sdo_gtype || ' srid=' || l_geometry.sdo_srid ||
    case when l_geometry.sdo_point is not null then
        ',x=' || l_geometry.sdo_point.x ||
        ',y=' || l_geometry.sdo_point.y ||
        ',z=' || l_geometry.sdo_point.z
    end ||
    ' )';
else
    l_value := apex_plugin_util.get_value_as_varchar2(p_data_type => l_column_value_list(col).data_type,
    p_value => l_column_value_list(col).value_list(row) );
end if;

sys.htp.p( case when col > 1 then ' - ' end || l_value );
end loop;

sys.htp.p('<li>');
end loop;
sys.htp.p('<ul>');

return null;
end;

29.14 GET_DATA2 Function Signature 2

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned along with their original data types. The search column is identified by providing a column number in the p_search_column_no parameter. This function takes into account character value comparison globalization attributes defines for the application.

Syntax

APEX_PLUGIN_UTIL.GET_DATA2 ( 
    p_sql_statement IN VARCHAR2,
    p_min_columns IN NUMBER,
    p_max_columns IN NUMBER,
    p_data_type_list IN WWV_GLOBAL.VC_ARR2 DEFAULT C_EMPTY_DATA_TYPE_LIST,
    p_component_name IN VARCHAR2,
    p_search_type IN VARCHAR2 DEFAULT 2,
p_search_column_name IN VARCHAR2 DEFAULT 2,
p_search_string IN VARCHAR2 DEFAULT NULL,
p_first_row IN NUMBER DEFAULT NULL,
p_max_rows IN NUMBER DEFAULT NULL)
RETURN t_column_value_list2;

Parameters

Table 29-16  GET_DATA2 Function Signature 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>SQL statement used for the lookup.</td>
</tr>
<tr>
<td>p_min_columns</td>
<td>Minimum number of return columns.</td>
</tr>
<tr>
<td>p_max_columns</td>
<td>Maximum number of return columns.</td>
</tr>
<tr>
<td>p_data_type_list</td>
<td>If provided, checks to make sure the data type for each column matches the specified data type in the array. Use the constants c_data_type_* for available data types.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>In case an error is returned, this is the name of the page item or report column used to display the error message.</td>
</tr>
<tr>
<td>p_search_type</td>
<td>Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore</td>
</tr>
<tr>
<td>p_search_column_name</td>
<td>The column name used to restrict the SQL statement.</td>
</tr>
<tr>
<td>p_search_string</td>
<td>Value used to restrict the query.</td>
</tr>
<tr>
<td>p_first_row</td>
<td>Start query at the specified row. All rows before the specified row are skipped.</td>
</tr>
<tr>
<td>p_max_rows</td>
<td>Maximum number of return rows allowed.</td>
</tr>
</tbody>
</table>

Return

Table 29-17  GET_DATA2 Function Signature 2 Return

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_column_value_list2</td>
<td>Table of t_column_values indexed by column number.</td>
</tr>
</tbody>
</table>

Example

The following example is a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list. This time, the first column of the LOV SQL statement is checked if it is of type VARCHAR2 and the second is of type number.

```java
function render_list ( 
    p_item    in apex_plugin.t_page_item,
    p_value   in varchar2,
    p_is_readonly in boolean,
    p_is_printer_friendly in boolean )
return apex_plugin.t_page_item_render_result
```

Chapter 29
GET_DATA2 Function Signature 2

29-16
is
  l_data_type_list apex_application_global.vc_arr2;
  l_column_value_list apex_plugin_util.t_column_value_list2;
begin
  -- The first LOV column has to be a string and the second a number
  l_data_type_list(1) := apex_plugin_util.c_data_type_varchar2;
  l_data_type_list(2) := apex_plugin_util.c_data_type_number;
  --
  l_column_value_list :=
    apex_plugin_util.get_data2 (p_sql_statement => p_item.lov_definition,
      p_min_columns      => 2,
      p_max_columns      => 2,
      p_data_type_list   => l_data_type_list,
      p_component_name   => p_item.name,
      p_search_type      => apex_plugin_util.c_search_contains_case,
      p_search_column_name => 'ENAME',
      p_search_string    => p_value );
  --
  sys.htp.p('<ul>');
  for i in 1 .. l_column_value_list.count(1)
    loop
    sys.htp.p('
      <li>' ||
        sys.htf.escape_sc(l_column_value_list(1).value_list(i).varchar2_value) ||
        -- display column
        ' - ' ||
        sys.htf.escape_sc(l_column_value_list(2).value_list(i).number_value) ||
        -- return column
        '</li>');
    end loop;
  sys.htp.p('</ul>');
end render_list;

29.15 GET_DISPLAY_DATA Function Signature 1

This function gets the display lookup value for the value specified in \textit{p\_search\_string}.

\textbf{Syntax}

\begin{verbatim}
APEX_PLUGIN_UTIL.GET_DISPLAY_DATA (p_sql_statement    IN VARCHAR2,
      p_min_columns      IN NUMBER,
      p_max_columns      IN NUMBER,
      p_component_name   IN VARCHAR2,
      p_display_column_no IN BINARY_INTEGER DEFAULT 1,
      p_search_column_no  IN BINARY_INTEGER DEFAULT 2,
      p_search_string    IN VARCHAR2 DEFAULT NULL,
      p_display_extra    IN BOOLEAN DEFAULT TRUE)
RETURN VARCHAR2;
\end{verbatim}
### Parameters

**Table 29-18  GET_DISPLAY_DATA Signature 1 Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>SQL statement used for the lookup.</td>
</tr>
<tr>
<td>p_min_columns</td>
<td>Minimum number of return columns.</td>
</tr>
<tr>
<td>p_max_columns</td>
<td>Maximum number of return columns.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>In case an error is returned, this is the name of the page item or report column used to display the error message.</td>
</tr>
<tr>
<td>p_display_column_no</td>
<td>Number of the column returned from the SQL statement. Must be within the p_min_columns though p_max_columns range.</td>
</tr>
<tr>
<td>p_search_column_no</td>
<td>Number of the column used to restrict the SQL statement. Must be within the p_min_columns though p_max_columns range.</td>
</tr>
<tr>
<td>p_search_string</td>
<td>Value used to restrict the query.</td>
</tr>
<tr>
<td>p_display_extra</td>
<td>If set to TRUE, and a value is not found, the search value is added to the result instead.</td>
</tr>
</tbody>
</table>

### Return

**Table 29-19  GET_DISPLAY_DATA Signature 1 Return**

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>Value of the first record of the column specified by p_display_column_no. If no record was found it contains the value of p_search_string if the parameter p_display_extra is set to TRUE. Otherwise NULL is returned.</td>
</tr>
</tbody>
</table>

### Example

The following example does a lookup with the value provided in p_value and returns the display column of the LOV query.

```java
function render_value (  
    p_item                in apex_plugin.t_page_item,  
    p_value               in varchar2,  
    p_is_readonly         in boolean,  
    p_is_printer_friendly in boolean )  
return apex_plugin.t_page_item_render_result  
is  
begin  
    sys.htp.p(sys.htf.escape_sc(  
        apex_plugin_util.get_display_data (  
            p_sql_statement     => p_item.lov_definition,  
            p_min_columns       => 2,  
            p_max_columns       => 2,  
            p_component_name    => p_item.name,  
            p_display_column_no => 1,  
            p_search_column_no  => 2,  
```
29.16 GET_DISPLAY_DATA Function Signature 2

This function looks up all the values provided in the p_search_value_list instead of just a single value lookup.

Syntax

APEX_PLUGIN_UTIL.GET_DISPLAY_DATA (
    p_sql_statement    IN VARCHAR2,
    p_min_columns      IN NUMBER,
    p_max_columns      IN NUMBER,
    p_component_name   IN VARCHAR2,
    p_display_column_no IN BINARY_INTEGER DEFAULT 1,
    p_search_column_no  IN BINARY_INTEGER DEFAULT 2,
    p_search_value_list IN ww_flow_global.vc_arr2,
    p_display_extra     IN BOOLEAN DEFAULT TRUE)
RETURN apex_application_global.vc_arr2;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>SQL statement used for the lookup.</td>
</tr>
<tr>
<td>p_min_columns</td>
<td>Minimum number of return columns.</td>
</tr>
<tr>
<td>p_max_columns</td>
<td>Maximum number of return columns.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>In case an error is returned, this is the name of the page item or report column used to display the error message.</td>
</tr>
<tr>
<td>p_display_column_no</td>
<td>Number of the column returned from the SQL statement. Must be within the p_min_columns through p_max_columns range.</td>
</tr>
<tr>
<td>p_search_column_no</td>
<td>Number of the column used to restrict the SQL statement. Must be within the p_min_columns through p_max_columns range.</td>
</tr>
<tr>
<td>p_search_value_list</td>
<td>Array of values to look up.</td>
</tr>
<tr>
<td>p_display_extra</td>
<td>If set to TRUE, and a value is not found, the search value is added to the result instead.</td>
</tr>
</tbody>
</table>
Table 29-21 GET_DISPLAY_DATA Signature 2 Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apex_application_global.vc_arr2</td>
<td>List of VARCHAR2 indexed by pls_integer. For each entry in p_search_value_list the resulting array contains the value of the first record of the column specified by p_display_column_no in the same order as in p_search_value_list. If no record is found it contains the value of p_search_string if the parameter p_display_extra is set to TRUE. Otherwise the value is skipped.</td>
</tr>
</tbody>
</table>

Example

Looks up the values 7863, 7911 and 7988 and generates a HTML list with the value of the corresponding display column in the LOV query.

```sql
function render_list (  
p_plugin in apex_plugin.t_plugin,  
p_item in apex_plugin.t_page_item,  
p_value in varchar2,  
p_is_readonly in boolean,  
p_is_printer_friendly in boolean )  
return apex_plugin.t_page_item_render_result  
is  
l_search_list apex_application_global.vc_arr2;  
l_result_list apex_application_global.vc_arr2;  
begin  
l_search_list(1) := '7863';  
l_search_list(2) := '7911';  
l_search_list(3) := '7988';  
--  
l_result_list :=  
apex_plugin_util.get_display_data (    
p_sql_statement => p_item.lov_definition,    
p_min_columns => 2,    
p_max_columns => 2,    
p_component_name => p_item.name,    
p_search_column_no => 1,    
p_search_value_list => l_search_list );  
--  
sys.htp.p('<ul>');  
for i in 1 .. l_result_list.count loop  
sys.htp.p('  ');  
sys.htp.escape_sc(l_result_list(i)) || '<li>';  
end loop;  
sys.htp.p('</ul>');  
end render_list;
```
29.17 GET_ELEMENT_ATTRIBUTES Function

This function returns some of the standard attributes of an HTML element (for example, id, name, required, placeholder, aria-error-attributes, class) which is used if a HTML input/select/textarea/... tag is generated to get a consistent set of attributes.

Syntax

APEX_PLUGIN_UTIL.GET_ELEMENT_ATTRIBUTES (
    p_item in apex_plugin.t_page_item,
    p_name in varchar2 default null,
    p_default_class in varchar2 default null,
    p_add_id in boolean default true,
    p_add_labelledby in boolean default true
    p_aria_describedby_id in varchar2 default null )
    return varchar2;

Parameters

Table 29-22    GET_ELEMENT_ATTRIBUTES Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item</td>
<td>This is the p_item parameter of your plug-in function.</td>
</tr>
<tr>
<td>p_name</td>
<td>This is the value which has been return by apex_plugin.get_input_name_or_page_item</td>
</tr>
<tr>
<td>p_default_class</td>
<td>Default CSS class which which should be contained in the result string.</td>
</tr>
<tr>
<td>p_add_id</td>
<td>If set to TRUE then the id attribute is also contained in the result string.</td>
</tr>
<tr>
<td>p_add_labelledby</td>
<td>Returns some of the general attributes of an HTML element (for example, the ID, name, required, placeholder, aria-error-attributes, class) which should be used if an HTML input, select, or textarea tag is generated to get a consistent set of attributes.Set to FALSE if you render a HTML input element like input, select, or textarea which does not require specifying the aria-labelledby attribute because the label's for attribute works for those HTML input elements. Set it to TRUE for all 'non-standard form element widgets (that is, those using div, span, and so on.) which do allow focus to make them accessible to screen readers.</td>
</tr>
<tr>
<td>p_aria_describedby_id</td>
<td>Pass additional IDs here that you would like get_element_attributes to include in the value it renders for the 'aria-describedby' attribute on the form element. This can be useful if you would like to convey additional information to users of Assistive Technology, when they are focused on the form field.</td>
</tr>
</tbody>
</table>

Note: Inclusion of aria-labelledby is also dependent on the item plug-in having Standard Form Element set to No and that there is a #LABEL_ID# substitution defined in the item's corresponding label template.
Example

This example emits an INPUT tag of type text which uses `apex_plugin_util.get_element_attributes` to automatically include the most common attributes.

```sql
sys.htp.prn ( ' <input type="text" ' ||
    apex_plugin_util.get_element_attributes(p_item, l_name,
    'text_field') ||
    ' value="'||l_escaped_value||'" '||
    ' size="'||p_item.element_width||'" '||
    'maxlength="'||p_item.element_max_length||'" '||
    ' />' );
```

29.18 GET_PLSQL_EXPRESSION_RESULT Function

This function executes a PL/SQL expression and returns a result. This function also performs the binding of any bind variables in the provided PL/SQL expression. This function is usually used for plug-in attributes of type PL/SQL Expression.

Syntax

```sql
APEX_PLUGIN_UTIL.GET_PLSQL_EXPRESSION_RESULT ( 
    p_plsql_expression IN VARCHAR2) 
RETURN VARCHAR2;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plsql_expression_result</td>
<td>A PL/SQL expression that returns a string.</td>
</tr>
</tbody>
</table>

Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>String result value returned by the PL/SQL Expression.</td>
</tr>
</tbody>
</table>

Example

This example executes and returns the result of the PL/SQL expression which is specified in attribute_03 of an item type plug-in attribute of type "PL/SQL Expression".

```sql
l_result := apex_plugin_util.get_plsql_expression_result ( 
    p_plsql_expression => p_item.attribute_03 );
```
29.19 GET_PLSQL_FUNCTION_RESULT Function

This function executes a PL/SQL function block and returns the result. This function also performs binding of bind variables in the provided PL/SQL Function Body. This function is usually used for plug-in attributes of type PL/SQL Function Body.

Syntax

APEX_PLUGIN_UTIL.GET_PLSQL_FUNCTION_RESULT (p_plsql_function IN VARCHAR2)
RETURN VARCHAR2;

Parameters

Table 29-25  GET_PLSQL_FUNCTION_RESULT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_plsql_function</td>
<td>A PL/SQL function block that returns a result of type string.</td>
</tr>
</tbody>
</table>

Return

Table 29-26  GET_PLSQL_FUNCTION_RESULT Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>String result value returned by the PL/SQL function block.</td>
</tr>
</tbody>
</table>

Example

The following example executes and returns the result of the PL/SQL function body that is specified in attribute_03 of an item type plug-in attribute of type PL/SQL Function Body.

l_result := apex_plugin_util.get_plsql_function_result (p_plsql_function => p_item.attribute_03);

29.20 GET_POSITION_IN_LIST Function

This function returns the position in the list where p_value is stored. If it is not found, null is returned.

Syntax

APEX_PLUGIN_UTIL.GET_POSITION_IN_LIST(p_list IN apex_application_global.vc_arr2, p_value IN VARCHAR2)
RETURN NUMBER;
Parameters

Table 29-27  GET_POSITION_IN_LIST Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_list</td>
<td>Array of type apex_application_global.vc_arr2 that contains entries of type VARCHAR2.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value located in the p_list array.</td>
</tr>
</tbody>
</table>

Return

Table 29-28  GET_POSITION_IN_LIST Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>Returns the position of p_value in the array p_list. If it is not found NULL is returned.</td>
</tr>
</tbody>
</table>

Example

The following example searches for "New York" in the provided list and returns 2 into l_position.

```sql
declare
    l_list     apex_application_global.vc_arr2;
    l_position number;
begin
    l_list(1) := 'Rome';
    l_list(2) := 'New York';
    l_list(3) := 'Vienna';

    l_position := apex_plugin_util.get_position_in_list (p_list => l_list, p_value => 'New York');
end;
```

29.21 GET_SEARCH_STRING Function

Based on the provided value in p_search_type the passed in value of p_search_string is returned unchanged or is converted to uppercase. Use this function with the p_search_string parameter of get_data and get_data2.

Syntax

```
APEX_PLUGIN_UTIL.GET_SEARCH_STRING(
    p_search_type IN VARCHAR2,
    p_search_string IN VARCHAR2)
RETURN VARCHAR2;
```
Parameters

Table 29-29  GET_SEARCH_STRING Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_search_type</td>
<td>Type of search when used with get_data and get_data2. Use one of the c_search_* constants.</td>
</tr>
<tr>
<td>p_search_string</td>
<td>Search string used for the search with get_data and get_data2.</td>
</tr>
</tbody>
</table>

Return

Table 29-30  GET_SEARCH_STRING Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>Returns p_search_string unchanged or in uppercase if p_search_type is of type c_search_contains_ignore or c_search_exact_ignore.</td>
</tr>
</tbody>
</table>

Example

This example uses a call to get_data or get_data2 to make sure the search string is using the correct case.

```
l_column_value_list :=
    apex_plugin_util.get_data (                    
        p_sql_statement    => p_item.lov_definition,       
        p_min_columns      => 2,                           
        p_max_columns      => 2,                           
        p_component_name   => p_item.name,                 
        p_search_type      => apex_plugin_util.c_search_contains_ignore, 
        p_search_column_no => 1,                           
        p_search_string    => apex_plugin_util.get_search_string (   
            p_search_type   => apex_plugin_util.c_search_contains_ignore, 
            p_search_string => p_value )));
```

29.22 GET_VALUE_AS_VARCHAR2 Function

This function can be used if you use GET_DATA2 to read the column values along with their original data types. It will convert and return the passed in p_value as VARCHAR2.

Syntax

```
function get_value_as_varchar2 (                    
    p_data_type in varchar2,               
    p_value in t_value,                    
    p_format_mask in varchar2 default null ) 
    return varchar2;
```
Parameters

Table 29-31  GET_VALUE_AS_VARCHAR2 Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_data_type</td>
<td>The data type of the value stored in p_value.</td>
</tr>
<tr>
<td>p_value</td>
<td>The value of type t_value which contains the value to be converted and returned as VARCHAR2.</td>
</tr>
<tr>
<td>p_format_mask</td>
<td>The format mask used to convert the value into a VARCHAR2.</td>
</tr>
</tbody>
</table>

Example

The following example emits all values stored in the data type aware l_column_value_list array as VARCHAR2.

```sql
declare
  l_column_value_list apex_plugin_util.t_column_value_list2;
begin
  -- Populate l_column_value_list by calling apex_plugin_util.get_data2
  ...
  -- Emit returned data
  sys.htp.p( '<table>' );
  for l_row in 1 .. l_column_value_list( 1 ).value_list.count loop
    sys.htp.p( '<tr>' );
    for l_column in 1 .. l_column_value_list.count loop
      sys.htp.p ( '<td>' ||
        apex_plugin_util.get_value_as_varchar2 ( p_data_type =>
          l_column_value_list( l_column ).data_type,
          p_value =>
          l_column_value_list( l_column ).value_list( l_row )
        ) ||
        '</td>' );
    end loop;
    sys.htp.p( '</tr>' );
  end loop;
  sys.htp.p( '</table>' );
end;
```

29.23 IS_EQUAL Function

This function returns TRUE if both values are equal and FALSE if not. If both values are NULL, TRUE is returned.
Syntax

APEX_PLUGIN_UTIL.IS_EQUAL (  
    p_value1 IN VARCHAR2  
    p_value2 IN VARCHAR2)  
RETURN BOOLEAN;

Parameters

Table 29-32  IS_EQUAL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value1</td>
<td>First value to compare.</td>
</tr>
<tr>
<td>p_value2</td>
<td>Second value to compare.</td>
</tr>
</tbody>
</table>

Return

Table 29-33  IS_EQUAL Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOLEAN</td>
<td>Returns TRUE if both values are equal or both values are NULL, otherwise it returns FALSE.</td>
</tr>
</tbody>
</table>

Example

In the following example, if the value in the database is different from what is entered, the code in the if statement is executed.

if NOT apex_plugin_util.is_equal(l_database_value, l_current_value) then
    -- value has changed, do something
    null;
end if;

29.24 PAGE_ITEM_NAMES_TO_JQUERY Function

This function returns a jQuery selector based on a comma delimited string of page item names. For example, you could use this function for a plug-in attribute called "Page Items to Submit" where the JavaScript code has to read the values of the specified page items.

Syntax

APEX_PLUGIN_UTIL.PAGE_ITEM_NAMES_TO_JQUERY (  
    p_page_item_names IN VARCHAR2)  
RETURN VARCHAR2;
Parameters

Table 29-34  PAGE_ITEM_NAMES_TO_JQUERY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_item_names</td>
<td>Comma delimited list of page item names.</td>
</tr>
</tbody>
</table>

Return

Table 29-35  PAGE_ITEM_NAMES_TO_JQUERY Return

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>Transforms the page items specified in p_page_item_names into a jQuery selector.</td>
</tr>
</tbody>
</table>

Example

The following example shows the code to construct the initialization call for a JavaScript function called myOwnWidget. This function gets an object with several attributes where one attribute is pageItemsToSubmit which is expected to be a jQuery selector.

```java
apex_javascript.add_onload_code {
    p_code => 'myOwnWidget('||
        '"'||p_item.name||'"',||
    '||
        apex_javascript.add_attribute('ajaxIdentifier',
            apex_plugin.get_ajax_identifier)||
        apex_javascript.add_attribute('dependingOnSelector',
            apex_plugin_util.page_item_names_to_jquery(p_item.lov_cascade_parent_items)
        )||
        apex_javascript.add_attribute('optimizeRefresh',
            p_item.ajax_optimize_refresh)||
        apex_javascript.add_attribute('pageItemsToSubmit',
            apex_plugin_util.page_item_names_to_jquery(p_item.ajax_items_to_submit))||
        apex_javascript.add_attribute('nullValue',
            p_item.lov_null_value, false, false)||
    '); ''
};
```

29.25 PRINT_DISPLAY ONLY Procedure

This procedure outputs a SPAN tag for a display only field.

Syntax

```sql
APEX_PLUGIN_UTIL.PRINT_DISPLAY_ONLY (
    p_item_name    IN VARCHAR2,
    p_display_value IN VARCHAR2,
    p_show_line_breaks IN BOOLEAN,
```
p_attributes       IN VARCHAR2,
p_id_postfix       IN VARCHAR2 DEFAULT '_DISPLAY');

Parameters

Table 29-36  PRINT_DISPLAYONLY Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item_name</td>
<td>Name of the page item. This parameter should be called with p_item.name.</td>
</tr>
<tr>
<td>p_display_value</td>
<td>Text to be displayed.</td>
</tr>
<tr>
<td>p_show_line_breaks</td>
<td>If set to TRUE line breaks in p_display_value are changed to &lt;br /&gt; so that the browser renders them as line breaks.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional attributes added to the SPAN tag.</td>
</tr>
<tr>
<td>p_id_postfix</td>
<td>Postfix which is getting added to the value in p_item_name to get the ID for the SPAN tag. Default is _DISPLAY.</td>
</tr>
</tbody>
</table>

Example

The following code could be used in an item type plug-in to render a display only page item.

```apex
apex_plugin_util.print_display_only {
    p_item_name     => p_item.name,
    p_display_value => p_value,
    p_show_line_breaks => false,
    p_escape        => true,
    p_attributes    => p_item.element_attributes
};
```

29.26 PRINT_ESCAPED_VALUE Procedure

This procedure outputs the value in an escaped form and chunks big strings into smaller outputs.

Syntax

```apex
APEX_PLUGIN_UTIL.PRINT_ESCAPED_VALUE (p_value IN VARCHAR2);
```

Parameters

Table 29-37  PRINT_ESCAPED_VALUE Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>Text which should be escaped and then printed to the HTTP buffer.</td>
</tr>
</tbody>
</table>
Example

Prints a hidden field with the current value of the page item.

sys.htp.prn('<input type="hidden" name="l_name" id="l_name" value=""');
print_escaped_value(p_value);
sys.htp.prn('">');

29.27 PRINT_HIDDEN_IF_READONLY Procedure

This procedure outputs a hidden field to store the page item value if the page item is rendered as readonly and is not printer friendly. If this procedure is called in an item type plug-in, the parameters of the plug-in interface should directly be passed in.

Syntax

APEX_PLUGIN_UTIL.PRINT_HIDDEN_IF_READONLY (  
    p_item_name   IN VARCHAR2,  
    p_value       IN VARCHAR2,  
    p_is_readonly IN BOOLEAN,  
    p_is_printer_friendly IN BOOLEAN,  
    p_id_postfix  IN VARCHAR2 DEFAULT NULL);

Parameters

Table 29-38  PRINT_HIDDEN_IF_READONLY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item_name</td>
<td>Name of the page item. For this parameter the p_item.name should be passed in.</td>
</tr>
<tr>
<td>p_value</td>
<td>Current value of the page item. For this parameter p_value should be passed in.</td>
</tr>
<tr>
<td>p_is_readonly</td>
<td>Is the item rendered readonly. For this parameter p_is_readonly should be passed in.</td>
</tr>
<tr>
<td>p_is_printer_friendly</td>
<td>Is the item rendered in printer friendly mode. For this parameter p_is_printer_friendly should be passed in.</td>
</tr>
<tr>
<td>p_id_postfix</td>
<td>Used to generate the ID attribute of the hidden field. It is build based on p_item_name and the value in p_id_postfix.</td>
</tr>
</tbody>
</table>

Example

Writes a hidden field with the current value to the HTTP output if p_is_readonly is TRUE and p_printer_friendly is FALSE.

apex_plugin_util.print_hidden_if_readonly (  
    p_item_name => p_item.name,  
    p_value => p_value,  
)
29.28 PRINT_JSON_HTTP_HEADER Procedure

This procedure outputs a standard HTTP header for a JSON output.

Syntax

APEX_PLUGIN_UTIL.PRINT_JSON_HTTP_HEADER;

Parameters

None.

Example

This example shows how to use this procedure in the Ajax callback function of a plugin. This code outputs a JSON structure in the following format: [{"d":"Display 1","r":"Return 1"},{"d":"Display 2","r":"Return 2"}]

-- Write header for the JSON stream.
APEX_PLUGIN_UTIL.PRINT_JSON_HTTP_HEADER;
-- initialize the JSON structure
sys.htp.p('{');
-- loop through the value array
for i in 1 .. l_values.count loop
  -- add array entry
  sys.htp.p (
    case when i > 1 then ', ' end||
    '{' ||
    apex/javascript.add_attribute('d',
      sys.htf.escape_sc(l_values(i).display_value), false, true)||
    apex/javascript.add_attribute('r',
      sys.htf.escape_sc(l_values(i).return_value), false, false)||
    '}');
end loop;
-- close the JSON structure
sys.htp.p('}');

29.29 PRINT_LOV_AS_JSON Procedure

This procedure outputs a JSON response based on the result of a two column LOV in the format:

[{"d":"display","r":"return"},{"d":......,"r":......},....]
Note:
The HTTP header is initialized with MIME type "application/json" as well.

Syntax

APEX_PLUGIN_UTIL.PRINT_LOV_AS_JSON (
    p_sql_statement         IN VARCHAR2,
    p_component_name        IN VARCHAR2,
    p_escape                IN BOOLEAN,
    p_replace_substitutions IN BOOLEAN DEFAULT FALSE);

Parameters

Table 29-39   PRINT_LOV_AS_JSON Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_sql_statement</td>
<td>A SQL statement which returns two columns from the SELECT.</td>
</tr>
<tr>
<td>p_component_name</td>
<td>The name of the page item or report column that is used in case an error is displayed.</td>
</tr>
<tr>
<td>p_escape</td>
<td>If set to TRUE the value of the display column is escaped, otherwise it is output as is.</td>
</tr>
<tr>
<td>p_replace_substitutions</td>
<td>If set to TRUE, apex_plugin_util.replace_substitutions is called for the value of the display column, otherwise, it is output as is.</td>
</tr>
</tbody>
</table>

Example

This example shows how to use the procedure in an Ajax callback function of an item type plug-in. The following call writes the LOV result as a JSON array to the HTTP output.

    apex_plugin_util.print_lov_as_json {
        p_sql_statement => p_item.lov_definition,
        p_component_name => p_item.name,
        p_escape         => true
    };

29.30 PRINT_OPTION Procedure

This procedure outputs an OPTION tag.

Syntax

APEX_PLUGIN_UTIL.PRINT_OPTION {
    p_display_value IN VARCHAR2,
    p_return_value IN VARCHAR2,
    p_is_selected IN BOOLEAN,
Parameters

Table 29-40  PRINT_OPTION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_display_value</td>
<td>Text which is displayed by the option.</td>
</tr>
<tr>
<td>p_return_value</td>
<td>Value which is set when the option is picked.</td>
</tr>
<tr>
<td>p_is_selected</td>
<td>Set to TRUE if the selected attribute should be set for this option.</td>
</tr>
<tr>
<td>p_attributes</td>
<td>Additional HTML attributes which should be set for the OPTION tag.</td>
</tr>
<tr>
<td>p_escape</td>
<td>Set to TRUE if special characters in p_display_value should be escaped.</td>
</tr>
</tbody>
</table>

Example

The following example could be used in an item type plug-in to create a SELECT list. Use apex_plugin_util.is_equal to find out which list entry should be marked as current.

```sql
sys.htp.p('<!-- loop through the result and add list entries -->
for i in 1 .. l_values.count loop
    apex_plugin_util.print_option (p_display_value => l_values(i).display_value,
                                   p_return_value  => l_values(i).return_value,
                                   p_is_selected   => apex_plugin_util.is_equal(l_values(i).return_value, p_value),
                                   p_attributes    => p_item.element_option_attributes,
                                   p_escape        => true);
end loop; sys.htp.p('</select>');
```

29.31 REPLACE_SUBSTITUTIONS Function

This function replaces any &ITEM. substitution references with their actual value. If p_escape is set to TRUE, any special characters contained in the value of the referenced item are escaped to prevent Cross-site scripting (XSS) attacks.

Syntax

```sql
apex_plugin_util.replace_substitutions (p_value in varchar2,
```

```sql
p_attributes IN VARCHAR2,
```
p_escape   in boolean default true )
return varchar2;

Parameters

**Table 29-41   REPLACE_SUBSTITUTION Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>This value is a string which can contain several &amp;ITEM. references which are replaced by their actual page item values.</td>
</tr>
<tr>
<td>p_escape</td>
<td>If set to TRUE any special characters contained in the value of the referenced item are escaped to prevent Cross-site scripting (XSS) attacks. If set to FALSE, the referenced items are not escaped.</td>
</tr>
</tbody>
</table>

Example

The following example replaces any substitution syntax references in the region plug-in attribute 05 with their actual values. Any special characters in the values are escaped.

```sql
l_advanced_formatting  := apex_plugin_util.replace_substitutions ( 
    p_value => p_region.attribute_05, 
    p_escape => true );
```

**29.32 SET_COMPONENT_VALUES Procedure**

This procedure extends Session State to include the column values of a specific row number. By doing so, columns can be referenced using substitution syntax or the \V{} function in the same way as you can reference page or application items.

> **Note:**

Always call `apex_plugin_util.clear_component_values` after you are done processing the current row!

Syntax

```sql
procedure set_component_values ( 
    p_column_value_list in t_column_list,
    p_row_num           in pls_integer );
```
Parameters

Table 29-42  SET_COMPONENT_VALUES Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_column_value_list</td>
<td>Table of t_column_values returned by the call to apex_plugin_util.get_data2.</td>
</tr>
<tr>
<td>p_row_num</td>
<td>Row number in p_column_value_list for which the column values should be set in Session State.</td>
</tr>
</tbody>
</table>

Example

This example is the skeleton of a simple item type plug-in rendering function which renders a link list based on a provided SQL query. Instead of a fixed SQL query format where the first column contains the link and the second contains the link label, it allows a developer using this plug-in to enter any SQL statement and then use substitution syntax to reference the values of the executed SQL query.

```plaintext
function render_link_list (p_item in apex_plugin.t_page_item, p_value in varchar2, p_is_readonly in boolean, p_is_printer_friendly in boolean) return apex_plugin.t_page_item_render_result is
    -- The link target plug-in attribute 01 would allow that a developer can enter a link which references columns of the provided SQL query using substitution syntax.
    -- For example: f?p=&APP_ID.:1:&APP_SESSION.:&DEBUG.::P1_EMPNO:&EMPNO.
    -- where &EMPNO. references the column EMPNO in the SQL query.
    c_link_target constant varchar2(4000) := p_item.attribute_01;
    -- The link label column plug-in attribute 02 would allow a developer to reference a column of the SQL query which should be used as the text for the link.
    c_link_label_column constant varchar2(128) := p_item.attribute_02;
    l_column_value_list apex_plugin_util.t_column_value_list2;
    begin
    l_column_value_list := apex_plugin_util.get_data2 (p_sql_statement => ...);
    sys.htp.p('<ul>');</
    for i in 1 .. l_column_value_list.count(1) loop
        -- Set all column values of the current row
        apex_plugin_util.set_component_values (p_column_value_list => l_column_value_list,
                                              p_row_num => i);
        --
        sys.htp.p(}
```
Chapter 29

SET_COMPONENT_VALUES Procedure

"<li><a href=" ||
apex_escape.html_attribute( apex_util.prepare_url( c_link_target )) ||
"">

apex_escape.html( v( c_link_label_column )) ||
</a></li>');

--
apex_plugin_util.clear_component_values;
end loop;
sys.htp.p('<ul>');
end;
The APEX_REGION package is the public API for handling regions.

- **IS_READ_ONLY Function**
- **PURGE_CACHE Procedure**

### 30.1 IS_READ_ONLY Function

This function returns TRUE if the current region is rendered read-only and FALSE if region is not rendered read-only. If the function is called from a context where no region is currently processed, it returns NULL. For example, you can use this function in conditions of a region or its underlying items and buttons.

**Syntax**

```sql
FUNCTION IS_READ_ONLY
RETURN BOOLEAN;
```

**Parameters**

None.

**Example**

This examples purges the session for a specific region cache for the whole application.

```sql
RETURN APEX_REGION.IS_READ_ONLY;
```

### 30.2 PURGE_CACHE Procedure

This procedure purges the region cache of the specified application, page, and region.

**Syntax**

```sql
PROCEDURE PURGE_CACHE (   p_application_id IN NUMBER DEFAULT apex.g_flow_id,   p_page_id IN NUMBER DEFAULT NULL,   p_region_id IN NUMBER DEFAULT NULL,   p_current_session_only IN BOOLEAN DEFAULT FALSE );
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>ID of the application where the region caches should be purged. Defaults to the current application.</td>
</tr>
<tr>
<td>p_page_id</td>
<td>ID of the page where the region caches should be purged. If no value is specified (which is the default), all regions of the application are purged.</td>
</tr>
<tr>
<td>p_region_id</td>
<td>ID of a specific region on a page. If no value is specified, all regions of the specified page are purged.</td>
</tr>
<tr>
<td>p_current_session_only</td>
<td>Specify true if you only want to purge entries that were saved for the current session. Defaults to false.</td>
</tr>
</tbody>
</table>

Example

This example purges session specific region cache for the whole application.

```
BEGIN
    APEX_REGION.PURGE_CACHE ( 
        p_current_session_only => true );
END;
```
## 31 APEX_SESSION

The package enables you to configure Application Express sessions.

- **SET_DEBUG Procedure**
- **SET_TRACE Procedure**
- **CREATE_SESSION Procedure**
- **DELETE_SESSION Procedure**
- **ATTACH Procedure**
- **DETACH Procedure**

### 31.1 SET_DEBUG Procedure

This procedure sets debug level for all future requests in a session.

**Syntax**

```sql
procedure set_debug (  
  p_session_id in number default apex.g_instance,  
  p_level in apex_debug_api.t_log_level );
```

**Parameters**

**Table 31-1  SET_DEBUG Procedure Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| p_session_id | The session id.  
**Note**: The session must belong to the current workspace or the caller must be able to set the session's workspace. |
| p_level      | The debug level. NULL disables debug, 1-9 sets a debug level.                |

**Example 1**

This example shows how to set debug for session 1234 to INFO level.

```sql
apex_session.set_debug (  
  p_session_id => 1234,  
  p_level => apex_debug.c_log_level_info );  
commit;
```
Example 2
This example shows how to disable debug in session 1234.

```apex_session.set_debug (  
    p_session_id => 1234,  
    p_level => null );
commit;
```

See Also:
- “ENABLE Procedure”
- “DISABLE Procedure”

31.2 SET_TRACE Procedure
This procedure sets trace mode in all future requests of a session.

Syntax

```procedure set_trace (  
    p_session_id in number default apex.g_instance,  
    p_mode in varchar2 );
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_session_id</td>
<td>The session id.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The session must belong to the current workspace or the caller must be able to set the session's workspace.</td>
</tr>
<tr>
<td>p_level</td>
<td>The trace mode. NULL disables trace, SQL enables SQL trace.</td>
</tr>
</tbody>
</table>

Example 1
This example shows how to enable trace in requests for session 1234.

```apex_session.set_trace (  
    p_session_id => 1234,  
    p_mode => 'SQL' );
commit;
```
Example 2

This example shows how to disable trace in requests for session 1234.

```apex_session.set_trace (
    p_session_id => 1234,
    p_mode => null );
commit;
```

31.3 CREATE_SESSION Procedure

This procedure creates a new session for the given application, set environment and run the application's Initialization PL/SQL Code.

Syntax

```procedure create_session (  
    p_app_id                   in number,  
    p_page_id                  in number,  
    p_username                 in varchar2,  
    p_call_post_authentication in boolean default false );
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>The application id.</td>
</tr>
<tr>
<td>p_page_id</td>
<td>The application page.</td>
</tr>
<tr>
<td>p_username</td>
<td>The session user.</td>
</tr>
<tr>
<td>p_call_post_authentication</td>
<td>If true, call post-authentication procedure. The default is false.</td>
</tr>
</tbody>
</table>

Raises

`WWV_FLOW.APP_NOT_FOUND_ERR`: The application does not exist or the caller has no access to the workspace.

Example

This example creates a session for EXAMPLE USER in application 100 page 1, then print the app id and session id.

```begin  
apex_session.create_session (  
    p_app_id   => 100,  
    p_page_id  => 1,  
    p_username => 'EXAMPLE USER' );  
sys.dbms_output.put_line (  
```
31.4 DELETE_SESSION Procedure

This procedure deletes the session with the given ID. If the session is currently attached, call the application's Cleanup PL/SQL Code and reset the environment. This procedure does nothing if the given session does not exist or if the caller can not access the session's workspace.

**Syntax**

```plaintext
procedure delete_session ( 
    p_session_id in number default wwv_flow.g_instance );
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_session_id</td>
<td>The session id.</td>
</tr>
</tbody>
</table>

**Raises**

- **APEX.SESSION.EXPIRED**: The session does not exist.
- **SECURITY_GROUP_ID_INVALID**: Current workspace does not match session workspace.

**Example**

Delete session 12345678.

```plaintext
begin 
    apex_session.delete_session ( 
        p_session_id => 12345678 );
end;
```
31.5 ATTACH Procedure

This procedure based on the given application and session current, sets environment and runs the Initialization PL/SQL Code.

Syntax

```plsql
procedure attach (  
    p_app_id     in number,  
    p_page_id    in number,  
    p_session_id in number );
```

Parameters

Table 31-5  Attach Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>The application id.</td>
</tr>
<tr>
<td>p_page_id</td>
<td>The application page.</td>
</tr>
<tr>
<td>p_session_id</td>
<td>The session id.</td>
</tr>
</tbody>
</table>

Raises

- **WWV_FLOW.APP_NOT_FOUND_ERR**: Application does not exist or caller has no access to the workspace.
- **APEX.SESSION.EXPIRED**: The session does not exist.
- **SECURITY_GROUP_ID_INVALID**: Current workspace does not match session workspace.

Example

Attach to session 12345678 for application 100 page 1, then print the app id and session id.

```plsql
begin  
    apex_session.attach (  
        p_app_id => 100,  
        p_page_id => 1,  
        p_session_id => 12345678 );  
    sys.dbms_output.put_line (  
```
31.6 DETACH Procedure

This procedure detaches from the current session, resets the environment and runs the application's Cleanup PL/SQL Code. This procedure does nothing if no session is attached.

Syntax

procedure detach;

Example

Detach from the current session..

begin
    apex_session.detach;
end;

See Also:

• "CREATE_SESSION Procedure"
• "DELETE_SESSION Procedure"
• "DETACH Procedure"
This package enables you to use Oracle Locator and the Spatial Option within Application Express. In an Application Express context, the logon user of the database session is typically APEX_PUBLIC_USER or ANONYMOUS. Spatial developers cannot directly use DML on USER_SDO_GEOM_METADATA within such a session, for example, in SQL Commands within SQL Workshop. The Spatial view’s trigger performs DML as the logon user, but it has to run as the application owner or workspace user. With the APEX_SPATIAL API, developers can use the procedures and functions below to insert, update and delete rows of USER_SDO_GEOM_METADATA as the current Application Express user. The package also provides a few utilities that simplify the use of Spatial in Application Express.

- Data Types
- CHANGE_GEOM_METADATA Procedure
- CIRCLE_POLYGON Function
- DELETE_GEOM_METADATA Procedure
- INSERT_GEOM_METADATA Procedure
- INSERT_GEOM_METADATA_LONLAT Procedure
- POINT Function
- RECTANGLE Function

### 32.1 Data Types

The data types used by this package are described in this section.

**t_srid**

subtype t_srid is number;

**c_no_reference_system**

c_no_reference_system constant t_srid := null;

**c_wgs_84**

c_wgs_84 constant t_srid := 4326; -- World Geodetic System, EPSG:4326

### 32.2 CHANGE_GEOM_METADATA Procedure

This procedure modifies a spatial metadata record.
Syntax

APEX_SPATIAL.CHANGE_GEOM_METADATA (  
  p_table_name        IN VARCHAR2,  
  p_column_name       IN VARCHAR2,  
  p_new_table_name    IN VARCHAR2 DEFAULT NULL,  
  p_new_column_name   IN VARCHAR2 DEFAULT NULL,  
  p_diminfo           IN mdsys.sdo_dim_array,  
  p_srid              IN t_srid );

Parameters

Table 32-1 CHANGE_GEOM_METADATA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of the feature table.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Name of the column of type mdsys.sdo_geometry.</td>
</tr>
<tr>
<td>p_new_table_name</td>
<td>New name of a feature table (or null, to keep the current value).</td>
</tr>
<tr>
<td>p_new_column_name</td>
<td>New name of the column of type mdsys.sdo_geometry (or null, to keep the current value).</td>
</tr>
<tr>
<td>p_diminfo</td>
<td>SDO_DIM_ELEMENT array, ordered by dimension, with one entry for each dimension.</td>
</tr>
<tr>
<td>p_srid</td>
<td>SRID value for the coordinate system for all geometries in the column.</td>
</tr>
</tbody>
</table>

Example

The code below modifies the dimensions of column CITIES.SHAPE.

begin
  for l_meta in ( select *  
                  from user_sdo_geom_metadata  
                  where table_name = 'CITIES'  
                    and column_name = 'SHAPE' )  
  loop
    apex.spatial.change_geom_metadata (  
      p_table_name => l_meta.table_name,  
      p_column_name => l_meta.column_name,  
      p_diminfo     => SDO_DIM_ARRAY (  
        SDO_DIM_ELEMENT('X', -180, 180, 0.1),  
        SDO_DIM_ELEMENT('Y', -90, 90, 0.1) ),  
      p_srid        => l_meta.srid );
  end loop;
end;
32.3 CIRCLE_POLYGON Function

This function creates a polygon that approximates a circle at \((p_{lon}, p_{lat})\) with radius of \(p_{radius}\). See mdsys.sdo_util.circle_polygon for details.

Syntax

APEX_SPATIAL.CIRCLE_POLYGON (  
  p_lon        IN NUMBER,  
  p_lat        IN NUMBER,  
  p_radius     IN NUMBER,  
  p_arc_tolerance  IN NUMBER DEFAULT 20,  
  p_srid        IN t_srid DEFAULT c_wgs_84 )  
RETURN mdsys.sdo_geometry;

Parameters

Table 32-2  CIRCLE_POLYGON Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lon</td>
<td>Longitude position of the lower left point.</td>
</tr>
<tr>
<td>p_lat</td>
<td>Latitude position of the lower left point.</td>
</tr>
<tr>
<td>p_radius</td>
<td>Radius of the circle in meters.</td>
</tr>
<tr>
<td>p_arc_tolerance</td>
<td>Arc tolerance (default 20).</td>
</tr>
<tr>
<td>p_srid</td>
<td>Reference system (default c_wgs_84).</td>
</tr>
</tbody>
</table>

Returns

Table 32-3  CIRCLE_POLYGON Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdsys.sdo_geometry</td>
<td>The geometry for the polygon that approximates the circle.</td>
</tr>
</tbody>
</table>

Example

This example is a query that returns a polygon that approximates a circle at \((0, 0)\) with radius 1.

```sql
select apex_spatial.circle_polygon(0, 0, 1) from dual
```

32.4 DELETE_GEOM_METADATA Procedure

This procedure deletes a spatial metadata record.
Syntax

APEX_SPATIAL.DELETE_GEOM_METADATA (  
  p_table_name        IN VARCHAR2,  
  p_column_name       IN VARCHAR2,  
  p_drop_index        IN BOOLEAN DEFAULT FALSE );

Parameters

Table 32-4  DELETE_GEOM_METADATA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of the feature table.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Name of the column of type mdsys.sdo_geometry.</td>
</tr>
<tr>
<td>p_drop_index</td>
<td>If TRUE (default is FALSE), drop the spatial index on the column.</td>
</tr>
</tbody>
</table>

Example

This example deletes metadata on column CITIES.SHAPE and drops the spatial index on this column.

begin
  apex_spatial.delete_geom_metadata (  
    p_table_name => 'CITIES',  
    p_column_name => 'SHAPE',  
    p_drop_index => true );
end;

32.5 INSERT_GEOM_METADATA Procedure

This procedure inserts a spatial metadata record and optionally creates a spatial index.

Syntax

APEX_SPATIAL.INSERT_GEOM_METADATA (  
  p_table_name        IN VARCHAR2,  
  p_column_name       IN VARCHAR2,  
  p_diminfo           IN mdsys.sdo_dim_array,  
  p_srid              IN t_srid,  
  p_create_index_name IN VARCHAR2 DEFAULT NULL );
### Parameters

#### Table 32-5  INSERT_GEOM_METADATA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>The name of the feature table.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>The name of the column of type mdsys.sdo_geometry.</td>
</tr>
<tr>
<td>p_diminfo</td>
<td>The SDO_DIM_ELEMENT array, ordered by dimension, with one entry for each dimension.</td>
</tr>
<tr>
<td>p_srid</td>
<td>The SRID value for the coordinate system for all geometries in the column.</td>
</tr>
<tr>
<td>p_create_index_name</td>
<td>If not null, a spatial index on the column is created with this name. Only simple column names are supported, function based indexes or indexes on object attributes cause an error. For more complex requirements, leave this parameter null (the default) and manually create the index.</td>
</tr>
</tbody>
</table>

#### Example

This example creates table CITIES, spatial metadata and an index on column CITIES.SHAPE.

```sql
create table cities (
  city_id   number primary key,
  city_name varchar2(30),
  shape     mdsys.sdo_geometry )
/
begin
  apex_spatial.insert_geom_metadata (   
    p_table_name   => 'CITIES',
    p_column_name  => 'SHAPE',
    p_diminfo     => SDO_DIM_ARRAY (   
      SDO_DIM_ELEMENT('X', -180, 180, 1),
      SDO_DIM_ELEMENT('Y',  -90, 90, 1) ),
    p_srid        => apex_spatial.c_wgs_84 );
end;
/
create index cities_idx_shape on cities(shape) indextype is mdsys.spatial_index
/
```

### 32.6 INSERT_GEOM_METADATA_LONLAT Procedure

This procedure inserts a spatial metadata record that is suitable for longitude/latitude and optionally creates a spatial index.

#### Syntax

```sql
APEX_SPATIAL.INSERT_GEOM_METADATA_LONLAT (   
  p_table_name   IN VARCHAR2,
```
p_column_name       IN VARCHAR2,
p_tolerance         IN NUMBER DEFAULT 1,
p_create_index_name IN VARCHAR2 DEFAULT NULL );

Parameters

Table 32-6 INSERT_GEOM_METADATA_LONLAT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of the feature table.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Name of the column of type mdsys.sdo_geometry.</td>
</tr>
<tr>
<td>p_tolerance</td>
<td>Tolerance value in each dimension, in meters (default 1).</td>
</tr>
<tr>
<td>p_create_index_name</td>
<td>if not null, a spatial index on the column is created with this name. Only simple column names are supported, function based indexes or indexes on object attributes cause an error. For more complex requirements, leave this parameter null (the default) and manually create the index.</td>
</tr>
</tbody>
</table>

Example

The code below creates table CITIES and spatial metadata for the column CITIES.SHAPE. By passing CITIES_IDX_SHAPE to p_create_index_name, the API call automatically creates an index on the spatial column.

```sql
create table cities (
    city_id   number primary key,
    city_name varchar2(30),
    shape     mdsys.sdo_geometry )
/
begin
    apex_spatial.insert_geom_metadata_lonlat (p_table_name => 'CITIES',
                                              p_column_name => 'SHAPE',
                                              p_create_index_name => 'CITIES_IDX_SHAPE');
end;
/
```

32.7 POINT Function

This function creates a point at \((p_{\text{lon}}, p_{\text{lat}})\).

Syntax

```sql
APEX_SPATIAL.POINT (p_lon IN NUMBER,
p_lat IN NUMBER,
p_srid IN t_srid DEFAULT c_wgs_84 )
RETURN mdsys.sdo_geometry;
```
Parameters

Table 32-7  POINT parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lon</td>
<td>Longitude position.</td>
</tr>
<tr>
<td>p_lat</td>
<td>Latitude position.</td>
</tr>
<tr>
<td>p_srid</td>
<td>Reference system (default c_wgs_84).</td>
</tr>
</tbody>
</table>

Returns

Table 32-8  POINT Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdsys.sdo_geometry</td>
<td>The geometry for the point.</td>
</tr>
</tbody>
</table>

Example

This example is a query that returns a point at (10, 50).

```sql
select apex_spatial.point(10, 50) from dual;
```

This example is equivalent to:

```sql
select mdsys.sdo_geometry(2001, 4326, sdo_point_type(10, 50, null), null, null) from dual;
```

32.8 RECTANGLE Function

This function creates a rectangle from point at \((p_{lon1}, p_{lat1})\) to \((p_{lon2}, p_{lat2})\).

Syntax

```sql
APEX_SPATIAL.RECTANGLE ( 
  p_lon1        IN NUMBER, 
  p_lat1        IN NUMBER, 
  p_lon2        IN NUMBER, 
  p_lat2        IN NUMBER, 
  p_srid        IN t_srid DEFAULT c_wgs_84 )
RETURN mdsys.sdo_geometry;
```

Parameters

Table 32-9  RECTANGLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lon1</td>
<td>Longitude position of the lower left point.</td>
</tr>
</tbody>
</table>
Table 32-9  (Cont.) RECTANGLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lat1</td>
<td>Latitude position of the lower left point.</td>
</tr>
<tr>
<td>p_lon2</td>
<td>Longitude position of the upper right point.</td>
</tr>
<tr>
<td>p_lat2</td>
<td>Latitude position of the upper right point.</td>
</tr>
<tr>
<td>p_srid</td>
<td>Reference system (default c_wgs_84).</td>
</tr>
</tbody>
</table>

Returns

Table 32-10  RECTANGLE Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdsys.sdo_geometry</td>
<td>The geometry for the rectangle (p_lon1, p_lon2, p_lon2, p_lat2).</td>
</tr>
</tbody>
</table>

Example

This example is a query that returns a rectangle from (10, 50) to (11, 51).

```
select apex_spatial.rectangle(10, 50, 11, 51) from dual
```

This example is equivalent to:

```
select mdsys.sdo_geometry(
    2003, 4326, null,
    sdo_elem_info_array(1, 1003, 1),
    sdo_ordinate_array(10, 50, 11, 50, 11, 51, 10, 51, 10, 50))
from dual;
```
The APEX_STRING package provides utilities for `varchar2`, `clob`, `apex_t_varchar2`, and `apex_t_number` types.

- **FORMAT Function**
- **GET_INITIALS Function**
- **GET_SEARCHABLE PhRASES Function**
- **GREP Function Signature 1**
- **GREP Function Signature 2**
- **GREP Function Signature 3**
- **JOIN_CLOB Function**
- **JOIN Function Signature 1**
- **JOIN Function Signature 2**
- **NEXT_CHUNK Function**
- **PLIST_DELETE Procedure**
- **PLIST_GET Function**
- **PLIST_PUSH Procedure**
- **PLIST_PUT Function**
- **PUSH Procedure Signature 1**
- **PUSH Procedure Signature 2**
- **PUSH Procedure Signature 3**
- **PUSH Procedure Signature 4**
- **SHUFFLE Function**
- **SHUFFLE Procedure**
- **SPLIT Function Signature 1**
- **SPLIT Function Signature 2**
- **SPLIT_NUMBERS Function**
- **STRING_TO_TABLE Function**
- **TABLE_TO_STRING Function**

### 33.1 FORMAT Function

Returns a formatted string, with substitutions applied.
Returns \texttt{p\_message} after replacing each \texttt{%s} with \texttt{p<n>} and each occurrence of \texttt{p<n>} with \texttt{p<n>}. If \texttt{p\_max\_length} is not null, \texttt{substr(p<n>)} is used instead of \texttt{p<n>}.

Syntax

\begin{verbatim}
format ( \\
p\_message \hspace{1em} \text{in varchar2}, \\
p0 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p1 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p2 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p3 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p4 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p5 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p6 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p7 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p8 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p9 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p10 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p11 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p12 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p13 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p14 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p15 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p16 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p17 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p18 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p19 \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null}, \\
p\_max\_length \hspace{1em} \text{in pls\_integer} \hspace{1em} \text{default 1000}, \\
p\_prefix \hspace{1em} \text{in varchar2} \hspace{1em} \text{default null} ) \\
return varchar2
\end{verbatim}

Parameters

Table 33-1 FORMAT Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{p_message}</td>
<td>Message string with substitution placeholders.</td>
</tr>
<tr>
<td>\texttt{p0-p19}</td>
<td>Substitution parameters.</td>
</tr>
<tr>
<td>\texttt{p_max_length}</td>
<td>If not null (default is 1000), cap each \texttt{p&lt;n&gt;} at \texttt{p_max_length} characters.</td>
</tr>
<tr>
<td>\texttt{p_prefix}</td>
<td>If set, remove leading white space and the given prefix from each line. This parameter can be used to simplify the formatting of indented multi-line text.</td>
</tr>
</tbody>
</table>

Example

\begin{verbatim}
apex\_string\_format('%s+%s=%s', 1, 2, 'three') \\
\rightarrow 1+2=three
\end{verbatim}
apex_string.format(©%1+%2=%0©, ©three©, 1, 2)
-> 1+2=three

apex_string.format ( 
  q!'begin
    !    if not valid then
    !        apex_debug.info('validation failed');
    !    end if;
    'end!','
  p_prefix => '!')
-> begin
    if not valid then
        apex_debug.info('validation failed');
    end if;
end;

33.2 GET_INITIALS Function

Get N letter initials from the first N words.

Syntax

get_initials ( 
    p_str in varchar2,
    p_cnt in number default 2 )
return varchar2

Parameters

Table 33-2  GET_INITIALS Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The input string.</td>
</tr>
<tr>
<td>p_cnt</td>
<td>The N letter initials to get from the first N words. The default is 2.</td>
</tr>
</tbody>
</table>

Example

Get initials from "John Doe".

begin
    sys.dbms_output.put_line(apex_string.get_initials('John Doe'));
end;
-> JD

begin
    sys.dbms_output.put_line(apex_string.get_initials(p_str => 'Andres Homero Lozano Garza', p_cnt => 3));
end;
-> AHL
33.3 GET_SEARCHABLE_PHRASES Function

This function returns distinct phrases of 1-3 consecutive lower case words in the input strings. Stopwords in the given language are ignored and split phrases.

**Note:**

This is a PL/SQL only implementation of a very small subset of what Oracle Text provides. Consider using Oracle Text instead, if the features and performance of this function are not sufficient.

**Syntax**

```plsql
function get_searchable_phrases (
    p_strings   in wwv_flow_t_varchar2,
    p_max_words in pls_integer default 3,
    p_language  in varchar2    default 'en' )
return wwv_flow_t_varchar2;
end wwv_flow_string;
```

**Parameters**

Table 33-3  GET_SEARCHABLE_PHRASES Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>The input string.</td>
</tr>
<tr>
<td>p_max_words</td>
<td>The maximum number of words in a phrase.</td>
</tr>
<tr>
<td></td>
<td>The default is 3.</td>
</tr>
<tr>
<td>p_language</td>
<td>The language identifier for stopwords, defaults to &quot;en&quot;. Supported values are &quot;cn&quot;,&quot;de&quot;,&quot;en&quot;,&quot;es&quot;,&quot;fr&quot;,&quot;it&quot;,&quot;ja&quot;,&quot;ko&quot;,&quot;pt-br&quot;.</td>
</tr>
</tbody>
</table>

**Example**

Prints keywords in the given input string.

```plsql
begin
    sys.dbms_output.put_line ( apex_string.join ( apex_string.get_searchable_phrases ( p_strings => apex_t_varchar2 ( 'Oracle APEX 19.1 is great.','Low code as it should be!' ))), 
                               ':
```
end;
Chapter 33
GREP Function Signature 1

33.4 GREP Function Signature 1

Returns the values of the input table that match a regular expression.

Syntax

grep (
    p_table         in apex_t_varchar2,
    p_pattern       in varchar2,
    p_modifier      in varchar2    default null,
    p_subexpression in varchar2    default '0',
    p_limit         in pls_integer default null )
return apex_t_varchar2;

Parameters

Table 33-4  GREP Function Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_pattern</td>
<td>The regular expression.</td>
</tr>
<tr>
<td>p_modifier</td>
<td>The regular expression modifier.</td>
</tr>
<tr>
<td>p_subexpression</td>
<td>The subexpression which should be returned. If null, return the complete</td>
</tr>
<tr>
<td></td>
<td>table value. If 0 (the default), return the matched expression. If &gt; 0,</td>
</tr>
<tr>
<td></td>
<td>return the subexpression value. You can also pass a comma separated list of</td>
</tr>
<tr>
<td></td>
<td>numbers, to get multiple subexpressions in the result.</td>
</tr>
<tr>
<td>p_limit</td>
<td>Limitation for the number of elements in the return table. If null (the</td>
</tr>
<tr>
<td></td>
<td>default), there is no limit.</td>
</tr>
</tbody>
</table>

Example

Collect and print basenames of sql files in input collection.

declare
    l_sqlfiles apex_t_varchar2;
begin
    l_sqlfiles := apex_string.grep (  
        p_table => apex_t_varchar2('a.html','b.sql',  
            'C.SQL'),
        p_pattern => '\w+\.sql',
        p_modifier => 'i',
        p_subexpression => '1' );
    sys.dbms_output.put_line(apex_string.join(l_sqlfiles, ':'));
end;

-> b:C
33.5 GREP Function Signature 2

Returns the values of the input `varchar2` that match a regular expression.

Syntax

```sql
grep (  
    p_str     in varchar2,  
    p_pattern in varchar2,  
    p_modifier in varchar2    default null,  
    p_subexpression in varchar2    default '0',  
    p_limit    in pls_integer default null )  
return apex_t_varchar2;
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_str</code></td>
<td>The input <code>varchar2</code>.</td>
</tr>
<tr>
<td><code>p_pattern</code></td>
<td>The regular expression.</td>
</tr>
<tr>
<td><code>p_modifier</code></td>
<td>The regular expression modifier.</td>
</tr>
<tr>
<td><code>p_subexpression</code></td>
<td>The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If &gt; 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.</td>
</tr>
<tr>
<td><code>p_limit</code></td>
<td>Limitation for the number of elements in the return table. If null (the default), there is no limit.</td>
</tr>
</tbody>
</table>

Example

Collect and print `key=value` definitions.

```sql
declare
    l_plist apex_t_varchar2;
begin
    l_plist := apex_string.grep (  
        p_str => 'define k1=v1'||chr(10)||
            'define k2 = v2',  
        p_pattern => 'define\s+(\w+)\s*\=\s*([^©||chr(10)||©]*)',  
        p_modifier => 'i',  
        p_subexpression => '1,2' );
    sys.dbms_output.put_line(apex_string.join(l_plist, ':'));
end;
-> k1:v1:k2:v2
```

33.6 GREP Function Signature 3

Returns the values of the input `clob` that match a regular expression.
Syntax

grep {
  p_str           in clob,
  p_pattern       in varchar2,
  p_modifier      in varchar2    default null,
  p_subexpression in varchar2    default '0',
  p_limit         in pls_integer default null }
return apex_t_varchar2;

Parameters

Table 33-6  GREP Function Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_str</td>
<td>The input clob.</td>
</tr>
<tr>
<td>p_pattern</td>
<td>The regular expression.</td>
</tr>
<tr>
<td>p_modifier</td>
<td>The regular expression modifier.</td>
</tr>
<tr>
<td>p_subexpression</td>
<td>The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If &gt; 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.</td>
</tr>
<tr>
<td>p_limit</td>
<td>Limitation for the number of elements in the return table. If null (the default), there is no limit.</td>
</tr>
</tbody>
</table>

Example

Collect and print key=value definitions.

declare
  l_plist apex_t_varchar2;
begin
  l_plist := apex_string.grep (p_str => to_clob('define k1=v1'||chr(10)||'define k2 = v2',
                                           p_pattern => 'define\s+\(\w+\)\s*=\s*\(\^[^\n]+\)',
                                           chr(10)||'\]'),
                            p_modifier => 'i',
                            p_subexpression => '1,2');
  sys.dbms_output.put_line(apex_string.join(l_plist, ':'));
end;
-> k1:v1:k2:v2

33.7 JOIN_CLOB Function

Returns the values of the apex_t_varchar2 input table p_table as a concatenated clob, separated by p_sep.
Syntax

```sql
join_clob (  
    p_table in apex_t_varchar2,  
    p_sep   in varchar2    default apex_application.LF,  
    p_dur   in pls_integer default sys.dbms_lob.call )  
return clob
```

Parameters

### Table 33-7  JOIN_CLOB Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator, default is line feed.</td>
</tr>
<tr>
<td>p_dur</td>
<td>The duration of the clob, default sys.dbms_lob.call.</td>
</tr>
</tbody>
</table>

Example

Concatenate numbers, separated by `':'`.

```sql
apex_string.join_clob(apex_t_varchar2(1,2,3), ':')
```

-> 1:2:3

### 33.8 JOIN Function Signature 1

Returns the values of the `apex_t_varchar2` input table `p_table` as a concatenated `varchar2`, separated by `p_sep`.

Syntax

```sql
join (  
    p_table in apex_t_varchar2,  
    p_sep in varchar2 default apex_application.LF)  
return varchar2
```

Parameters

### Table 33-8  JOIN Function Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator, default is line feed.</td>
</tr>
</tbody>
</table>
Example

Concatenate numbers, separated by `:`.

```
apex_string.join(apex_t_varchar2('a','b','c'),':')
-> a:b:c
```

33.9 JOIN Function Signature 2

Returns the values of the `apex_t_number` input table `p_table` as a concatenated `varchar2`, separated by `p_sep`.

Syntax

```
join (  
    p_table in apex_t_number,  
    p_sep in varchar2 default apex_application.LF )  
return varchar2
```

Parameters

**Table 33-9  JOIN Function Signature 2 Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator, default is line feed.</td>
</tr>
</tbody>
</table>

Example

Concatenate numbers, separated by `:`.

```
apex_string.join(apex_t_number(1,2,3),':')
-> 1:2:3
```

33.10 NEXT_CHUNK Function

This function reads a fixed-length string from a clob. This is just a small wrapper around `DBMS_LOB.READ`, however it prevents common errors when incrementing the offset and picking the maximum chunk size.

Syntax

```
function next_chunk (  
    p_str in            clob,  
    p_chunk out         nocopy varchar2,  
    p_offset in out nocopy pls_integer,  
    p_amount in         pls_integer default 8191 )  
return boolean;
```
Parameters

Table 33-10 NEXT CHUNK Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_str</td>
<td>The input clob.</td>
</tr>
<tr>
<td>p_chunk</td>
<td>The chunk value (in/out).</td>
</tr>
<tr>
<td>p_offset</td>
<td>The position in p_str, where the next chunk should be read from (in/out).</td>
</tr>
<tr>
<td>p_amount</td>
<td>The amount of characters that should be read (default 8191).</td>
</tr>
</tbody>
</table>

Returns

True if another chunk could be read. False if reading past the end of p_str.

Example

Print chunks of 25 bytes of the input clob.

```
declare
  l_input  clob := 'The quick brown fox jumps over the lazy dog';
  l_offset pls_integer;
  l_chunk  varchar2(20);
begin
  while apex_string.next_chunk (p_str => l_input,
                               p_chunk => l_chunk,
                               p_offset => l_offset,
                               p_amount => 20 )
  loop
    sys.dbms_output.put_line(l_chunk);
  end loop;
end;
```

Output:
The quick brown fox
jumps over the lazy
dog

33.11 PLIST_DELETE Procedure

This procedure removes the property list key from the table.

Syntax

```
plist_delete (p_table in out nocopy apex_t_varchar2,
              p_key    in varchar2 );
```
Parameters

Table 33-11  PLIST_DELETE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_key</td>
<td>The input key.</td>
</tr>
</tbody>
</table>

Raised Errors

Table 33-12  PLIST_DELETE Procedure Raised Errors

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_DATA_FOUND</td>
<td>Given key does not exist in table.</td>
</tr>
</tbody>
</table>

Example

Remove value of property "key2".

declare
   l_plist apex_t_varchar2 :=
   apex_t_varchar2('key1','foo','key2','bar');
begin
   apex_string.plist_delete(l_plist,'key2');
   sys.dbms_output.put_line(apex_string.join(l_plist,':'));
end;
-> key1:foo

33.12 PLIST_GET Function

This function gets the property list value for a key.

Syntax

plist_get (  
   p_table in apex_t_varchar2,  
   p_key in varchar2  
)  
return varchar2

Parameters

Table 33-13  PLIST_GET Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_key</td>
<td>The input key.</td>
</tr>
</tbody>
</table>
Raised Errors

Table 33-14  PLIST_GET Function Raised Errors

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_DATA_FOUND</td>
<td>Given key does not exist in table.</td>
</tr>
</tbody>
</table>

Example

Get value of property "key2".

declare
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
begin
    sys.dbms_output.put_line(apex_string.plist_get(l_plist,'key2'));
end;
-> bar

33.13 PLIST_PUSH Procedure

This procedure appends key/value to the property list, without looking for duplicates.

Syntax

procedure plist_push (  
p_table in out nocopy wwv_flow_t_varchar2,  
p_key   in varchar2,  
p_value in varchar2 );

Parameters

Table 33-15  PLIST_PUSH Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_key</td>
<td>The input key.</td>
</tr>
<tr>
<td>p_value</td>
<td>The input value.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to append key2/bar.

declare
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo');
begin
    apex_string.plist_push(l_plist,'key2','bar');
    sys.dbms_output.put_line(apex_string.plist_get(l_plist,'key2'));
end;
33.14 PLIST_PUT Function

This function inserts or updates property list value for a key.

Syntax

```plaintext
plist_put(
    p_table in out nocopy apex_t_varchar2,
    p_key   in varchar2,
    p_value in varchar2);
```

Parameters

Table 33-16  PLIST_PUT Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
<tr>
<td>p_key</td>
<td>The input key.</td>
</tr>
<tr>
<td>p_value</td>
<td>The input value.</td>
</tr>
</tbody>
</table>

Example

Set property value to "key2".

```plaintext
declare
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo');
begin
    apex_string.plist_put(l_plist,'key2','bar');
    sys.dbms_output.put_line(apex_string.plist_get(l_plist,'key2'));
end;
```

33.15 PUSH Procedure Signature 1

This procedure appends value to apex_t_varchar2 table.

Syntax

```plaintext
push(
    p_table in out nocopy apex_t_varchar2,
    p_value in varchar2);
```
Parameters

Table 33-17  PUSH Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>Defines the table.</td>
</tr>
<tr>
<td>p_value</td>
<td>Specifies the value to be added.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to append 2 values, then prints the table.

```sql
declare
  l_table apex_t_varchar2;
begin
  apex_string.push(l_table, 'a');
  apex_string.push(l_table, 'b');
  sys.dbms_output.put_line(apex_string.join(l_table, ':'));
end;
-> a:b
```

33.16 PUSH Procedure Signature 2

This procedure appends a value to apex_t_number table.

Syntax

```sql
push (    
  p_table in out nocopy apex_t_number,    
  p_value in number );
```

Parameters

Table 33-18  PUSH Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>Defines the table.</td>
</tr>
<tr>
<td>p_value</td>
<td>Specifies the value to be added.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to append 2 values, then prints the table.

```sql
declare
  l_table apex_t_number;
begin
  apex_string.push(l_table, 1);
  apex_string.push(l_table, 2);
  sys.dbms_output.put_line(apex_string.join(l_table, ':'));
```
33.17 PUSH Procedure Signature 3

This procedure appends collection values to apex_t_varchar2 table.

Syntax

push (  
p_table  in out nocopy apex_t_varchar2,  
p_values in            apex_t_varchar2 );

Parameters

Table 33-19  PUSH Procedure Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>Defines the table.</td>
</tr>
<tr>
<td>p_values</td>
<td>Specifies the values that should be added to</td>
</tr>
<tr>
<td></td>
<td>p_table.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to append a single value and multiple values, then prints the table.

declare  
  l_table apex_t_varchar2;  
begin  
  apex_string.push(l_table, 'a');  
  apex_string.push(l_table, apex_t_varchar2('1','2','3'));  
  sys.dbms_output.put_line(apex_string.join(l_table, ':'));  
end;  
-> a:1:2:3

33.18 PUSH Procedure Signature 4

This procedure appends values of a PL/SQL table to apex_t_varchar2 table.

Syntax

procedure push (  
p_table  in out nocopy wwv_flow_t_varchar2,  
p_values in            wwv_flow_global.vc_arr2 );
### Parameters

#### Table 33-20  PUSH Procedure Signature 4 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>Defines the table.</td>
</tr>
<tr>
<td>p_values</td>
<td>Specifies the values that should be added to p_table.</td>
</tr>
</tbody>
</table>

### Example

The following example demonstrates how to append the values of a PL/SQL table, then prints the table.

```sql
declare
    l_table  apex_t_varchar2;
    l_values apex_application_global.vc_arr2;
begin
    l_values(1) := 'a';
    l_values(2) := 'b';
    apex_string.push(l_table, l_values);
    sys.dbms_output.put_line(apex_string.join(l_table, ':'));
end;
```

-> a:b

### 33.19 SHUFFLE Function

Returns the input table values, re-ordered.

#### Syntax

```sql
shuffle (
    p_table in apex_t_varchar2
) return apex_t_varchar2;
```

#### Parameters

#### Table 33-21  SHUFFLE Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table.</td>
</tr>
</tbody>
</table>

#### Example

Shuffle and print l_table.

```sql
declare
    l_table apex_t_varchar2 := apex_string.split('1234567890',null);
begin
    ...
```

```sql
...```
33.20 SHUFFLE Procedure

This procedure randomly re-orders the values of the input table.

Syntax

```
shuffle (  
    p_table  in out nocopy apex_t_varchar2 );
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table, which will be modified by the procedure.</td>
</tr>
</tbody>
</table>

Example

Shuffle and print l_table.

```
declare
    l_table apex_t_varchar2 := apex_string.split('1234567890',null);
begin
    apex_string.shuffle(l_table);
    sys.dbms_output.put_line(apex_string.join(l_table,':'));
end;
-> a permutation of 1:2:3:4:5:6:7:8:9:0
```

33.21 SPLIT Function Signature 1

Use this function to split input string at separator.

Syntax

```
split (  
    p_str   in varchar2,  
    p_sep   in varchar2 default apex_application.LF,  
    p_limit in pls_integer default null )  
return apex_t_varchar2;
```
Parameters

Table 33-23  SPLIT Function Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_str</td>
<td>The input string.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression. The default is to split at line feed.</td>
</tr>
<tr>
<td>p_limit</td>
<td>Maximum number of splits, ignored if null. If smaller than the total possible number of splits, the last table element contains the rest.</td>
</tr>
</tbody>
</table>

Examples

```
apex_string.split(1||chr(10)||2||chr(10)||3)
-> apex_t_varchar2('©1©,©2©,©3©)

apex_string.split('1:2:3',':')
-> apex_t_varchar2('©1©,©2©,©3©)

apex_string.split('123',null)
-> apex_t_varchar2('©1©,©2©,©3©')

apex_string.split('1:2:3:4',':',2)
-> apex_t_varchar2('©1©,©2:3:4©')

apex_string.split('key1=val1, key2=val2','\s*[=,]\s*')
-> apex_t_varchar2('key1', 'val1', 'key2', 'val2')
```

33.22 SPLIT Function Signature 2

Use this function to split input clob at separator.

Syntax

```
split (  
  p_str in clob,  
  p_sep in varchar2 default apex_application.LF  )  
return apex_t_varchar2;
```

Parameters

Table 33-24  SPLIT Function Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_str</td>
<td>The input clob.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression. The default is to split at line feed</td>
</tr>
</tbody>
</table>
Example

apex_string.split('1:2:3',':')
-> apex_t_varchar2('1','2','3')

33.23 SPLIT_NUMBERS Function

Use this function to split input at separator, values must all be numbers.

Syntax

split_numbers ( 
    p_str in varchar2,
    p_sep in varchar2 default apex_application.LF )
return apex_t_number;

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_str</td>
<td>The input varchar2.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression. The default is to split at line feed.</td>
</tr>
</tbody>
</table>

Example

apex_string.split_numbers('1:2:3',':')
-> apex_t_number(1,2,3)

33.24 STRING_TO_TABLE Function

Returns the split input at separator, returning a vc_arr2.

Syntax

function string_to_table ( 
    p_str in varchar2,
    p_sep in varchar2 default ':'
) 
return wwv_flow_global.vc_arr2;
Parameters

Table 33-26 STRING_TO_TABLE Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_str</td>
<td>The input varchar2.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator, no regexp or split at char. Defaults to ':'.</td>
</tr>
</tbody>
</table>

Example

```sql
declare
  l_result apex_application_global.vc_arr2;
begin
  l_result := apex_string.string_to_table('1:2:3',':');
  sys.dbms_output.put_line(apex_string.table_to_string(l_result,'-'));
end;
```
```
1-2-3
```

33.25 TABLE_TO_STRING Function

This function returns the values of the apex_application_global.vc_arr2 input table `p_table` as a concatenated varchar2, separated by `p_sep`.

Syntax

```sql
function table_to_string (  
  p_table in wwv_flow_global.vc_arr2,  
  p_sep   in varchar2             default ':')  
return varchar2;
```

Parameters

Table 33-27 TABLE_TO_STRING Function Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table</td>
<td>The input table, assumes no holes and index starts at 1.</td>
</tr>
<tr>
<td>p_sep</td>
<td>The separator, default is ':'.</td>
</tr>
</tbody>
</table>

Example

Concatenate numbers, separated by ':'.

```sql
declare
  l_table apex_application_global.vc_arr2;
begin
  l_table(1) := 'a';
  l_table(2) := 'b';
  l_table(3) := 'c';
```
sys.dbms_output.put_line(apex_string.table_to_string(l_table));
end;
-> a:b:c
The APEX_THEME package contains utility functions for working with themes and theme styles.

- CLEAR_ALL_USERS_STYLE Procedure
- CLEAR_USER_STYLE Procedure
- DISABLE_USER_STYLE Procedure
- ENABLE_USER_STYLE Procedure
- GET_USER_STYLE Function
- SET_CURRENT_STYLE Procedure
- SET_SESSION_STYLE Procedure
- SET_SESSION_STYLE_CSS Procedure
- SET_USER_STYLE Procedure

### 34.1 CLEAR_ALL_USERS_STYLE Procedure

This procedure clears all theme style user preferences for an application and theme.

**Syntax**

```sql
procedure clear_all_users_style(   
    p_application_id  IN NUMBER           DEFAULT {current application id},
    p_theme_number    IN NUMBER           DEFAULT {current theme id});
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application to clear all user theme style preferences for.</td>
</tr>
<tr>
<td>p_theme_number</td>
<td>The theme number to clear all theme style user preferences for.</td>
</tr>
</tbody>
</table>
Example

The following example clears the all theme style user preferences for theme 42 in application 100.

```sql
apex_theme.clear_all_users_style(
    p_application_id => 100,
    p_theme_number => 42
);
```

34.2 CLEAR_USER_STYLE Procedure

This procedure clears the theme style user preference for user and application.

Syntax

```sql
procedure clear_user_style(
    p_application_id IN NUMBER          DEFAULT {current application id},
    p_user           IN VARCHAR2        DEFAULT {current user},
    p_theme_number   IN  NUMBER         DEFAULT {current theme number}
);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_theme_number</td>
<td>The theme number to clear the theme style user preference.</td>
</tr>
</tbody>
</table>

Example

The following example clears the theme style user preference for the ADMIN user in application 100 and theme 42.

```sql
apex_theme.clear_user_style(
    p_application_id => 100,
    p_user => 'ADMIN',
    p_theme_number => 42
);
```

34.3 DISABLE_USER_STYLE Procedure

This procedure disables theme style selection by end users. End users will not be able to customize the theme style on their own. Note that this only affects the Customization link for end users. APEX_THEME API calls are independent.
Syntax

```sql
procedure disable_user_style(
    p_application_id  IN NUMBER            DEFAULT {current application id},
    p_theme_number    IN NUMBER            DEFAULT {current theme number});
```

Parameters

Table 34-3  DISABLE_USER_STYLE Procedure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The Application ID.</td>
</tr>
<tr>
<td>p_theme_number</td>
<td>Number of User Interface's <em>Current Theme</em>.</td>
</tr>
</tbody>
</table>

The following example disable end user theme style selection for the Desktop user interface of application 100.

```sql
declare
    l_theme_id apex_themes.theme_number%type;
begin
    select theme_number into l_theme_id
    from apex_appl_user_interfaces
    where application_id = 100
    and display_name   = 'Desktop';

    apex_theme.disable_user_style(
        p_application_id => 100,
        p_theme_number   => l_theme_id
    );
end;
```

34.4 ENABLE_USER_STYLE Procedure

This procedure enables theme style selection by end users. When enabled and there is at least one theme style marked as Public, end users will see a Customize link which allows to choose the theme style. End user theme style selection is enabled or disabled at the User Interface level. When providing a theme number, the theme must be the *Current Theme* for a user interface. Note that this only affects the Customization link for end users. APEX_THEME API calls are independent.

Syntax

```sql
procedure enable_user_style(
    p_application_id  IN NUMBER            DEFAULT {current application id},
    p_theme_number    IN NUMBER            DEFAULT {current theme number});
```
Parameters

Table 34-4  ENABLE_USER_STYLE Procedure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The Application ID.</td>
</tr>
<tr>
<td>p_theme_number</td>
<td>Number of User Interface’s Current Theme.</td>
</tr>
</tbody>
</table>

The following example enable end user theme style selection for the Desktop user interface of application 100.

```sql
declare
    l_theme_id apex_themes.theme_number%type;
begin
    select theme_number into l_theme_id
    from apex_appl_user_interfaces
    where application_id = 100
    and display_name = 'Desktop';

    apex_theme.enable_user_style(
        p_application_id => 100,
        p_theme_number   => l_theme_id
    );
end;
```

34.5 GET_USER_STYLE Function

This function returns the theme style user preference for the user and application. If no user preference is present, NULL is returned.

Syntax

```sql
function get_user_style(
    p_application_id  IN NUMBER           DEFAULT {current application id},
    p_user            IN VARCHAR2         DEFAULT {current user},
    p_theme_number    IN NUMBER           DEFAULT {current theme number}
) return number;
```

Parameters

Table 34-5  GET_USER_STYLE Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application to set the user style preference.</td>
</tr>
<tr>
<td>p_user</td>
<td>The user name to the user style preference.</td>
</tr>
<tr>
<td>p_theme_number</td>
<td>The theme number to set the session style.</td>
</tr>
</tbody>
</table>
Table 34-5  (Cont.) GET_USER_STYLE Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN</td>
<td>The theme style ID which is set as a user preference.</td>
</tr>
</tbody>
</table>

Example

The query returns the theme style user preference for the ADMIN user in application 100 and theme 42.

```sql
select apex_theme.get_user_style( 100, 'ADMIN', 42 ) from dual;
```

34.6 SET_CURRENT_STYLE Procedure

This procedure sets current theme style for the current application.

Syntax

```sql
procedure set_current_style (  
    p_theme_number  IN NUMBER,  
    p_id            IN VARCHAR2  
);  
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_theme_number</td>
<td>The theme number for which to set the default style.</td>
</tr>
<tr>
<td>p_style_id</td>
<td>The ID of the new default theme style.</td>
</tr>
</tbody>
</table>

Example

The following example gets available theme styles from Application Express Dictionary View for the DESKTOP user interface.

```sql
select s.theme_style_id, t.theme_number  
from apex_application_theme_styles s,  
apex_application_themes t  
where s.application_id = t.application_id  
    and s.theme_number = t.theme_number  
    and s.application_id = :app_id  
    and t.ui_type_name = 'DESKTOP'  
    and s.ls_current = 'Yes'
```
The following example sets the current theme style to one of values returned by the above query.

```sql
apex_theme.set_current_style(
    p_theme_number => query.theme_number,
    p_id => query.theme_style_id
);
```

See Also:

"SET_CURRENT_THEME_STYLE Procedure [DEPRECATED]"

34.7 SET_SESSION_STYLE Procedure

This procedure sets the theme style dynamically for the current session. This is typically being called after successful authentication.

Syntax

```sql
procedure set_session_style (  
    p_theme_number  IN NUMBER           DEFAULT (current theme number),  
    p_name          IN VARCHAR2
);
```

Parameters

Table 34-7   SET_SESSION_STYLE Procedure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_theme_number</td>
<td>The theme number to set the session style for, default is the current theme of the application.</td>
</tr>
<tr>
<td>p_name</td>
<td>The name of the theme style to be used in the session.</td>
</tr>
</tbody>
</table>

Example

The following example gets the current theme number from Application Express Dictionary View for the DESKTOP user interface.

```sql
select t.theme_number  
    from apex_application_themes t  
where t.application_id = :app_id  
    and t.ui_type_name = 'DESKTOP'
```
The following example sets the session theme style for the current theme to Vita.

```sql
apex_theme.set_current_style (
    p_theme_number => {query.theme_number},
    p_name => 'Vita'
);
```

### 34.8 SET_SESSION_STYLE_CSS Procedure

This procedure sets the theme style CSS urls dynamically for the current session. Theme style CSS URLs are being directly passed in; a persistent style definition is not needed. This is typically being called after successful authentication.

**Syntax**

```sql
procedure set_session_style_css (
    p_theme_number  IN NUMBER           DEFAULT {current theme number},
    p_css_urls     IN VARCHAR2
);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_theme_number</td>
<td>The theme number to set the session style.</td>
</tr>
<tr>
<td>p_css_urls</td>
<td>The URLs to CSS files with style directives.</td>
</tr>
</tbody>
</table>

**Example**

The following example gets available theme styles from Application Express Dictionary View for the DESKTOP user interface.

```sql
select s.theme_style_id, t.theme_number
from apex_application_theme_styles s,
     apex_application_themes t
where s.application_id = t.application_id
and s.theme_number = t.theme_number
and s.application_id = :app_id
and t.ui_type_name = 'DESKTOP'
and s.is_current = 'Yes'
```

The following example sets the current theme style to one of values returned by the above query.

```sql
apex_theme.set_session_style_css(
    p_theme_number => {query.theme_number},
    p_css_urls => {URLs to theme style CSS files}
);
```
34.9 SET_USER_STYLE Procedure

This procedure sets a theme style user preference for the current user and application. Theme Style User Preferences are automatically picked up and precede any style set with SET_SESSION_STYLE.

Syntax

```
procedure set_user_style(
    p_application_id  IN NUMBER           DEFAULT  {current application id},
    p_user            IN VARCHAR2         DEFAULT  {current user},
    p_theme_number    IN NUMBER           DEFAULT  {current theme number},
    p_id              IN NUMBER
);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application to set the user style preference.</td>
</tr>
<tr>
<td>p_user</td>
<td>The user name to the user style preference.</td>
</tr>
<tr>
<td>p_theme_number</td>
<td>The theme number to set the user style preference.</td>
</tr>
<tr>
<td>p_id</td>
<td>The ID of the theme style to set as a user preference.</td>
</tr>
</tbody>
</table>

Example

The following example gets available theme styles from Application Express Dictionary View for the DESKTOP user interface.

```
select s.theme_style_id, t.theme_number
    from apex_application_theme_styles s,
    apex_application_themes t
    where s.application_id = t.application_id
        and s.theme_number = t.theme_number
        and s.application_id = :app_id
        and t.ui_type_name = 'DESKTOP'
        and s.is_current = 'Yes'
```

The following example sets the current theme style id's as user preference for ADMIN in application ID 100.

```
apex_theme.set_user_style (
    p_application_id => 100,  
    p_user           => 'ADMIN',
    p_theme_number   => {query.theme_number},
)
Chapter 34

SET_USER_STYLE Procedure

```sql

p_id => (query.theme_style_id)
```

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.

- **ADD_AD_COLUMN Procedure**
- **ADD_AD_SYNONYM Procedure**
- **DEL_AD_COLUMN Procedure**
- **DEL_AD_SYNONYM Procedure**
- **DEL_COLUMN Procedure**
- **DEL_GROUP Procedure**
- **DEL_TABLE Procedure**
- **SYNCH_TABLE Procedure**
- **UPD_AD_COLUMN Procedure**
- **UPD_AD_SYNONYM Procedure**
- **UPD_COLUMN Procedure**
- **UPD_DISPLAY_IN_FORM Procedure**
- **UPD_DISPLAY_IN_REPORT Procedure**
- **UPD_FORM_REGION_TITLE Procedure**
- **UPD_GROUP Procedure**
- **UPD_ITEM_DISPLAY_HEIGHT Procedure**
- **UPD_ITEM_DISPLAY_WIDTH Procedure**
- **UPD_ITEM_FORMAT_MASK Procedure**
- **UPD_ITEM_HELP Procedure**
- **UPD_LABEL Procedure**
- **UPD_REPORT_ALIGNMENT Procedure**
- **UPD_REPORT_FORMAT_MASK Procedure**
- **UPD_REPORT_REGION_TITLE Procedure**
35.1 ADD_AD_COLUMN Procedure

Adds a User Interface Default Attribute Dictionary entry with the provided definition. Up to three synonyms can be provided during the creation. Additional synonyms can be added post-creation using apex_ui_default_update.add_ad_synonym. Synonyms share the column definition of their base column.

Syntax

APEX_UI_DEFAULT_UPDATE.ADD_AD_COLUMN (  
  p_column_name IN VARCHAR2,  
  p_label IN VARCHAR2 DEFAULT NULL,  
  p_help_text IN VARCHAR2 DEFAULT NULL,  
  p_format_mask IN VARCHAR2 DEFAULT NULL,  
  p_default_value IN VARCHAR2 DEFAULT NULL,  
  p_form_format_mask IN VARCHAR2 DEFAULT NULL,  
  p_form_display_width IN VARCHAR2 DEFAULT NULL,  
  p_form_display_height IN VARCHAR2 DEFAULT NULL,  
  p_form_data_type IN VARCHAR2 DEFAULT NULL,  
  p_report_format_mask IN VARCHAR2 DEFAULT NULL,  
  p_report_col_alignment IN VARCHAR2 DEFAULT NULL,  
  p_syn_name1 IN VARCHAR2 DEFAULT NULL,  
  p_syn_name2 IN VARCHAR2 DEFAULT NULL,  
  p_syn_name3 IN VARCHAR2 DEFAULT NULL);

Parameters

Table 35-1 ADD_AD_COLUMN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_column_name</td>
<td>Name of column to be created.</td>
</tr>
<tr>
<td>p_label</td>
<td>Used for item label and report column heading.</td>
</tr>
<tr>
<td>p_help_text</td>
<td>Used for help text for items and interactive report columns</td>
</tr>
<tr>
<td>p_format_mask</td>
<td>Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.</td>
</tr>
<tr>
<td>p_default_value</td>
<td>Used as the default value for items.</td>
</tr>
<tr>
<td>p_form_format_mask</td>
<td>If provided, used as the format mask for items, overriding any value for the general format mask.</td>
</tr>
<tr>
<td>p_form_display_width</td>
<td>Used as the width of any items using this Attribute Definition.</td>
</tr>
</tbody>
</table>
Table 35-1  (Cont.) ADD_AD_COLUMN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_form_display_height</td>
<td>Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).</td>
</tr>
<tr>
<td>p_form_data_type</td>
<td>Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.</td>
</tr>
<tr>
<td>p_report_format_mask</td>
<td>If provided, used as the format mask for report columns, overriding any value for the general format mask.</td>
</tr>
<tr>
<td>p_report_col_alignment</td>
<td>Used as the alignment for report column data (for example, number are usually right justified). Valid values are LEFT, CENTER, and RIGHT.</td>
</tr>
<tr>
<td>p_syn_name1</td>
<td>Name of synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.</td>
</tr>
<tr>
<td>p_syn_name2</td>
<td>Name of second synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.</td>
</tr>
<tr>
<td>p_syn_name3</td>
<td>Name of third synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.</td>
</tr>
</tbody>
</table>

Example

The following example creates a new attribute to the UI Defaults Attribute Dictionary within the workspace associated with the current schema. It also creates a synonym for that attribute.

BEGIN
  apex_ui_default_update.add_ad_column (  
    p_column_name          => 'CREATED_BY',  
    p_label                => 'Created By',  
    p_help_text            => 'User that created the record.',  
    p_form_display_width   => 30,  
    p_form_data_type       => 'VARCHAR',  
    p_report_col_alignment => 'LEFT',  
    p_syn_name1            => 'CREATED_BY_USER' );
END;

35.2 ADD_AD_SYNONYM Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the synonym provided is created and associated with that column. Synonyms share the column definition of their base column.
**Syntax**

APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM (
    p_column_name IN VARCHAR2,
    p_syn_name    IN VARCHAR2);

**Parameters**

**Table 35-2 ADD_AD_SYNONYM Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_column_name</td>
<td>Name of column with the Attribute Dictionary that the synonym is being created for.</td>
</tr>
<tr>
<td>p_syn_name</td>
<td>Name of synonym to be created.</td>
</tr>
</tbody>
</table>

**Example**

The following example add the synonym CREATED_BY_USER to the CREATED_BY attribute of the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

BEGIN
    apex_ui_default_update.add_ad_synonym ( 
        p_column_name => 'CREATED_BY',
        p_syn_name    => 'CREATED_BY_USER' );
END;

### 35.3 DEL_AD_COLUMN Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the column, along with any associated synonyms, is deleted.

**Syntax**

APEX_UI_DEFAULT_UPDATE.DEL_AD_COLUMN ( 
    p_column_name IN VARCHAR2);

**Parameters**

**Table 35-3 DEL_AD_COLUMN Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_column_name</td>
<td>Name of column to be deleted</td>
</tr>
</tbody>
</table>
Example

The following example deletes the attribute CREATED_BY from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

BEGIN
    apex_ui_default_update.del_ad_column (
        p_column_name => 'CREATED_BY' );
END;

35.4 DEL_AD_SYNONYM Procedure

If the synonym name is found within the User Interface Default Attribute Dictionary, the synonym name is deleted.

Syntax

APEX_UI_DEFAULT_UPDATE.DEL_AD_SYNONYM ( 
    p_syn_name           IN VARCHAR2);

Parameters

Table 35-4  DEL_AD_SYNONYM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_syn_name</td>
<td>Name of synonym to be deleted</td>
</tr>
</tbody>
</table>

Example

The following example deletes the synonym CREATED_BY_USER from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

BEGIN
    apex_ui_default_update.del_ad_synonym ( 
        p_syn_name => 'CREATED_BY_USER' );
END;

35.5 DEL_COLUMN Procedure

If the provided table and column exists within the user’s schema's table based User Interface Defaults, the UI Defaults for it are deleted.

Syntax

APEX_UI_DEFAULT_UPDATE.DEL_COLUMN ( 
    p_table_name            IN VARCHAR2,  
    p_column_name           IN VARCHAR2);
Parameters

Table 35-5  DEL_COLUMN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of table whose column's UI Defaults are to be deleted.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Name of columns whose UI Defaults are to be deleted.</td>
</tr>
</tbody>
</table>

Example

The following example deletes the column `CREATED_BY` from the `EMP` table definition within the UI Defaults Table Dictionary within the current schema.

BEGIN
    apex_ui_default_update.del_column (  
        p_table_name  => 'EMP',  
        p_column_name => 'CREATED_BY'  
    );
END;

35.6 DEL_GROUP Procedure

If the provided table and group exists within the user's schema's table based User Interface Defaults, the UI Defaults for it are deleted and any column within the table that references that group has the group_id set to null.

Syntax

APEX_UI_DEFAULT_UPDATE.DEL_GROUP (  
    p_table_name            IN VARCHAR2,  
    p_group_name            IN VARCHAR2);  

Parameters

Table 35-6  DEL_GROUP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of table whose group UI Defaults are to be deleted</td>
</tr>
<tr>
<td>p_group_name</td>
<td>Name of group whose UI Defaults are to be deleted</td>
</tr>
</tbody>
</table>

Example

The following example deletes the group `AUDIT_INFO` from the `EMP` table definition within the UI Defaults Table Dictionary within the current schema.

BEGIN
    apex_ui_default_update.del_group (  
        p_table_name => 'EMP',  
        p_group_name => 'AUDIT_INFO'  
    );
END;
35.7 DEL_TABLE Procedure

If the provided table exists within the user’s schema’s table based User Interface Defaults, the UI Defaults for it is deleted. This includes the deletion of any groups defined for the table and all the columns associated with the table.

Syntax

APEX_UI_DEFAULT_UPDATE.DEL_TABLE (  
    p_table_name   IN VARCHAR2);

Parameters

Table 35-7  DEL_TABLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
</tbody>
</table>

Example

The following example removes the UI Defaults for the EMP table that are associated with the current schema.

```sql
begin
    apex_ui_default_update.del_table (  
        p_table_name => 'EMP' );
end;
/
```

35.8 SYNCH_TABLE Procedure

If the Table Based User Interface Defaults for the table do not already exist within the user’s schema, they are defaulted. If they do exist, they are synchronized, meaning, the columns in the table is matched against the column in the UI Defaults Table Definitions. Additions and deletions are used to make them match.

Syntax

APEX_UI_DEFAULT_UPDATE.SYNCH_TABLE (  
    p_table_name   IN VARCHAR2);
Parameters

Table 35-8  SYNCH_TABLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
</tbody>
</table>

Example

The following example synchronizes the UI Defaults for the EMP table that are associated with the current schema.

BEGIN
    apex_ui_default_update.synch_table ( p_table_name => 'EMP' );
END;

35.9 UPD_AD_COLUMN Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the column entry is updated using the provided parameters. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_AD_COLUMN ( p_column_name           IN  VARCHAR2,  
                                        p_new_column_name       IN  VARCHAR2  DEFAULT NULL,  
                                        p_label                 IN  VARCHAR2  DEFAULT NULL,  
                                        p_help_text             IN  VARCHAR2  DEFAULT NULL,  
                                        p_format_mask           IN  VARCHAR2  DEFAULT NULL,  
                                        p_default_value         IN  VARCHAR2  DEFAULT NULL,  
                                        p_form_format_mask      IN  VARCHAR2  DEFAULT NULL,  
                                        p_form_display_width    IN  VARCHAR2  DEFAULT NULL,  
                                        p_form_display_height   IN  VARCHAR2  DEFAULT NULL,  
                                        p_form_data_type        IN  VARCHAR2  DEFAULT NULL,  
                                        p_report_format_mask    IN  VARCHAR2  DEFAULT NULL,  
                                        p_report_col_alignment  IN  VARCHAR2  DEFAULT NULL );

Parameters

Table 35-9  UPD_AD_COLUMN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_column_name</td>
<td>Name of column to be updated</td>
</tr>
<tr>
<td>p_new_column_name</td>
<td>New name for column, if column is being renamed</td>
</tr>
<tr>
<td>p_label</td>
<td>Used for item label and report column heading</td>
</tr>
<tr>
<td>p_help_text</td>
<td>Used for help text for items and interactive report columns</td>
</tr>
</tbody>
</table>
Table 35-9  (Cont.) UPD_AD_COLUMN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_format_mask</td>
<td>Used as the format mask for items and report columns. Can be overwritten by</td>
</tr>
<tr>
<td></td>
<td>report for form specific format masks.</td>
</tr>
<tr>
<td>p_default_value</td>
<td>Used as the default value for items.</td>
</tr>
<tr>
<td>p_form_format_mask</td>
<td>If provided, used as the format mask for items, overriding any value for</td>
</tr>
<tr>
<td></td>
<td>the general format mask.</td>
</tr>
<tr>
<td>p_form_display_width</td>
<td>Used as the width of any items using this Attribute Definition.</td>
</tr>
<tr>
<td>p_form_display_height</td>
<td>Used as the height of any items using this Attribute Definition (only used</td>
</tr>
<tr>
<td></td>
<td>by item types such as text areas and shuttles).</td>
</tr>
<tr>
<td>p_form_data_type</td>
<td>Used as the data type for items (results in an automatic validation).</td>
</tr>
<tr>
<td></td>
<td>Valid values are VARCHAR, NUMBER and DATE.</td>
</tr>
<tr>
<td>p_report_format_mask</td>
<td>If provided, used as the format mask for report columns, overriding any</td>
</tr>
<tr>
<td></td>
<td>value for the general format mask.</td>
</tr>
<tr>
<td>p_report_col_alignment</td>
<td>Used as the alignment for report column data (for example, number are</td>
</tr>
<tr>
<td></td>
<td>usually right justified). Valid values are LEFT, CENTER, and RIGHT.</td>
</tr>
</tbody>
</table>

**Note:**

If p_label through p_report_col_alignment are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

**Example**

The following example updates the CREATED_BY column in the UI Defaults Attribute Dictionary within the workspace associated with the current schema, setting the form_format_mask to null.

```sql
BEGIN
    apex_ui_default_update.upd_ad_column (
        p_column_name => 'CREATED_BY',
        p_form_format_mask => 'null%');
END;
```

35.10 UPD_AD_SYNONYM Procedure

If the synonym name is found within the User Interface Default Attribute Dictionary, the synonym name is updated.
Syntax

APEX_UI_DEFAULT_UPDATE.UPD_AD_SYNONYM (  
    p_syn_name           IN VARCHAR2,  
    p_new_syn_name       IN VARCHAR2 DEFAULT NULL);

Parameters

Table 35-10    UPD_AD_SYNONYM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_syn_name</td>
<td>Name of synonym to be updated</td>
</tr>
<tr>
<td>p_new_syn_name</td>
<td>New name for synonym</td>
</tr>
</tbody>
</table>

Example

The following example updates the `CREATED_BY_USER` synonym in the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

BEGIN
    apex_ui_default_update.upd_ad_synonym (  
        p_syn_name     => 'CREATED_BY_USER',  
        p_new_syn_name => 'USER_CREATED_BY');
END;

35.11 UPD_COLUMN Procedure

If the provided table and column exists within the user's schema's table based User Interface Defaults, the provided parameters are updated. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_COLUMN (  
    p_table_name            IN VARCHAR2,  
    p_column_name           IN VARCHAR2,  
    p_group_id              IN VARCHAR2  DEFAULT NULL,  
    p_label                 IN VARCHAR2  DEFAULT NULL,  
    p_help_text             IN VARCHAR2  DEFAULT NULL,  
    p_display_in_form       IN VARCHAR2  DEFAULT NULL,  
    p_display_seq_form      IN VARCHAR2  DEFAULT NULL,  
    p_mask_form             IN VARCHAR2  DEFAULT NULL,  
    p_default_value         IN VARCHAR2  DEFAULT NULL,  
    p_required              IN VARCHAR2  DEFAULT NULL,  
    p_display_width         IN VARCHAR2  DEFAULT NULL,  
    p_max_width             IN VARCHAR2  DEFAULT NULL,  
    p_height                IN VARCHAR2  DEFAULT NULL,  
    p_display_in_report     IN VARCHAR2  DEFAULT NULL,  
    p_display_seq_report    IN VARCHAR2  DEFAULT NULL,  
);
### Parameters

#### Table 35-11  UPD_COLUMN Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_table_name</code></td>
<td>Name of table whose column's UI Defaults are being updated</td>
</tr>
<tr>
<td><code>p_column_name</code></td>
<td>Name of column whose UI Defaults are being updated</td>
</tr>
<tr>
<td><code>p_group_id</code></td>
<td>id of group to be associated with the column</td>
</tr>
<tr>
<td><code>p_label</code></td>
<td>When creating a form against this table or view, this is used as the label for the item if this column is included. When creating a report or tabular form, this is used as the column heading if this column is included.</td>
</tr>
<tr>
<td><code>p_help_text</code></td>
<td>When creating a form against this table or view, this becomes the help text for the resulting item.</td>
</tr>
<tr>
<td><code>p_display_in_form</code></td>
<td>When creating a form against this table or view, this determines whether this column is displayed in the resulting form page. Valid values are Y and N.</td>
</tr>
<tr>
<td><code>p_display_seq_form</code></td>
<td>When creating a form against this table or view, this determines the sequence in which the columns is displayed in the resulting form page.</td>
</tr>
<tr>
<td><code>p_mask_form</code></td>
<td>When creating a form against this table or view, this specifies the mask that is applied to the item, such as 999-99-9999. This is not used for character based items.</td>
</tr>
<tr>
<td><code>p_default_value</code></td>
<td>When creating a form against this table or view, this specifies the default value for the item resulting from this column.</td>
</tr>
<tr>
<td><code>p_required</code></td>
<td>When creating a form against this table or view, this specifies to generate a validation in which the resulting item must be NOT NULL. Valid values are Y and N.</td>
</tr>
<tr>
<td><code>p_display_width</code></td>
<td>When creating a form against this table or view, this specifies the display width of the item resulting from this column.</td>
</tr>
<tr>
<td><code>p_max_width</code></td>
<td>When creating a form against this table or view, this specifies the maximum string length that a user is allowed to enter in the item resulting from this column.</td>
</tr>
<tr>
<td><code>p_height</code></td>
<td>When creating a form against this table or view, this specifies the display height of the item resulting from this column.</td>
</tr>
<tr>
<td><code>p_display_in_report</code></td>
<td>When creating a report against this table or view, this determines whether this column is displayed in the resulting report. Valid values are Y and N.</td>
</tr>
<tr>
<td><code>p_display_seq_report</code></td>
<td>When creating a report against this table or view, this determines the sequence in which the columns are displayed in the resulting report.</td>
</tr>
<tr>
<td><code>p_mask_report</code></td>
<td>When creating a report against this table or view, this specifies the mask that is applied against the data, such as 999-99-9999. This is not used for character based items.</td>
</tr>
<tr>
<td><code>p_alignment</code></td>
<td>When creating a report against this table or view, this determines the alignment for the resulting report column. Valid values are L for Left, C for Center, and R for Right.</td>
</tr>
</tbody>
</table>
Note:

If p_group_id through p_alignment are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the column DEPT_NO within the EMP table definition within the UI Defaults Table Dictionary within the current schema, setting the group_id to null.

BEGIN
    apex_ui_default_update.upd_column (p_table_name => 'EMP',
        p_column_name => 'DEPT_NO',
        p_group_id => 'null%');
END;

35.12 UPD_DISPLAY_IN_FORM Procedure

The UPD_DISPLAY_IN_FORM procedure sets the display in form user interface defaults. This user interface default is used by wizards when you select to create a form based upon the table. It controls whether the column is included by default or not.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM (p_table_name => VARCHAR2,
    p_column_name => VARCHAR2,
    p_display_in_form => VARCHAR2);

Parameters

Table 35-12  UPD_DISPLAY_IN_FORM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_display_in_form</td>
<td>Determines whether to display in the form by default, valid values are Y and N</td>
</tr>
</tbody>
</table>

Example

In the following example, when creating a Form against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM (p_table_name => 'DEPT',
35.13 UPD_DISPLAY_IN_REPORT Procedure

The UPD_DISPLAY_IN_REPORT procedure sets the display in report user interface default. This user interface default is used by wizards when you select to create a report based upon the table and controls whether the column is included by default or not.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_REPORT (  
  p_table_name IN VARCHAR2,  
  p_column_name IN VARCHAR2,  
  p_display_in_report IN VARCHAR2);  

Parameters

Table 35-13 UPD_DISPLAY_IN_REPORT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_display_in_report</td>
<td>Determines whether to display in the report by default, valid values are Y and N</td>
</tr>
</tbody>
</table>

Example

In the following example, when creating a Report against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

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

35.14 UPD_FORM_REGION_TITLE Procedure

The UPD_FORM_REGION_TITLE procedure updates the Form Region Title user interface default. User interface defaults are used in wizards when you create a form based upon the specified table.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_FORM_REGION_TITLE (  
  p_table_name IN VARCHAR2,  
  p_form_region_title IN VARCHAR2 DEFAULT NULL);
**Parameters**

Table 35-14  UPDATE_FORM_REGION_TITLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_form_region_title</td>
<td>Desired form region title</td>
</tr>
</tbody>
</table>

**Example**

This example demonstrates how to set the Forms Region Title user interface default on the DEPT table.

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

### 35.15 UPD_GROUP Procedure

If the provided table and group exist within the user's schema's table based User Interface Defaults, the group name, description and display sequence of the group are updated. If 'null%' is passed in for p_description or p_display_sequence, the value is set to null.

**Syntax**

APEX_UI_DEFAULT_UPDATE.UPD_GROUP (  
  p_table_name            IN VARCHAR2,  
  p_group_name            IN VARCHAR2,  
  p_new_group_name        IN VARCHAR2 DEFAULT NULL,  
  p_description           IN VARCHAR2 DEFAULT NULL,  
  p_display_sequence      IN VARCHAR2 DEFAULT NULL);

**Parameters**

Table 35-15  UPD_GROUP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of table whose group is being updated</td>
</tr>
<tr>
<td>p_group_name</td>
<td>Group being updated</td>
</tr>
<tr>
<td>p_new_group_name</td>
<td>New name for group, if group is being renamed</td>
</tr>
<tr>
<td>p_description</td>
<td>Description of group</td>
</tr>
<tr>
<td>p_display_sequence</td>
<td>Display sequence of group.</td>
</tr>
</tbody>
</table>
Note:

If `p_description` or `p_display_sequence` are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the description of the group `AUDIT_INFO` within the `EMP` table definition within the UI Defaults Table Dictionary within the current schema.

```sql
BEGIN
    apex_ui_default_update.upd_group (
        p_table_name  => 'EMP',
        p_group_name  => 'AUDIT_INFO',
        p_description => 'Audit columns' );
END;
```

35.16 UPD_ITEM_DISPLAY_HEIGHT Procedure

The `UPD_ITEM_DISPLAY_HEIGHT` procedure sets the item display height user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column. Display height controls if the item is a text box or a text area.

Syntax

```sql
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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_display_height</td>
<td>Display height of any items created based upon this column</td>
</tr>
</tbody>
</table>

Example

The following example sets a default item height of 3 when creating an item on the `DNAME` column against the `DEPT` table.

```sql
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT(
    p_table_name => 'DEPT',
```
35.17 UPD_ITEM_DISPLAY_WIDTH Procedure

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

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_WIDTH (  
  p_table_name            IN VARCHAR2,  
  p_column_name           IN VARCHAR2,  
  p_display_width         IN NUMBER);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_display_width</td>
<td>Display width of any items created based upon this column</td>
</tr>
</tbody>
</table>

Example

The following example sets a default item width of 5 when creating an item on the DEPTNO column against the DEPT table.

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

35.18 UPD_ITEM_FORMAT_MASK Procedure

The UPD_ITEM_FORMAT_MASK procedure sets the item format mask user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column. Item format mask is typically used to format numbers and dates.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_ITEM_FORMAT_MASK (  
  p_table_name            IN VARCHAR2,  
  p_column_name           IN VARCHAR2,  
  p_format_mask           IN VARCHAR2 DEFAULT NULL);
Parameters

Table 35-18  UPD_ITEM_FORMAT_MASK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_format_mask</td>
<td>Format mask to be associated with the column</td>
</tr>
</tbody>
</table>

Example

In the following example, when creating a Form against the EMP table, the default item format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

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

35.19 UPD_ITEM_HELP Procedure

The UPD_ITEM_HELP procedure updates the help text for the specified table and column. This user interface default is used when you create a form based upon the table and select to include the specified column.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_help_text             IN VARCHAR2 DEFAULT NULL);

Parameters

Table 35-19  UPD_ITEM_HELP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_help_text</td>
<td>Desired help text</td>
</tr>
</tbody>
</table>

Example

This example demonstrates how to set the User Interface Item Help Text default for the DEPTNO column in the DEPT table.

APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP(
    p_table_name => 'DEPT',
...
35.20 UPD_LABEL Procedure

The UPD_LABEL procedure sets the label used for items. This user interface default is used when you create a form or report based on the specified table and include a specific column.

Syntax

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

Parameters

Table 35-20  UPD_LABEL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_label</td>
<td>Desired item label</td>
</tr>
</tbody>
</table>

Example

This example demonstrates how to set the User Interface Item Label default for the DEPTNO column in the DEPT table.

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

35.21 UPD_REPORT_ALIGNMENT Procedure

The UPD_REPORT_ALIGNMENT procedure sets the report alignment user interface default. This user interface default is used by wizards when you select to create a report based upon the table and include the specified column and determines if the report column should be left, center, or right justified.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_REPORT_ALIGNMENT (  
    p_table_name            IN VARCHAR2,  
    p_column_name           IN VARCHAR2,  
    p_report_alignment      IN VARCHAR2);
Parameters

Table 35-21  UPD_REPORT_ALIGNMENT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name.</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name.</td>
</tr>
<tr>
<td>p_report_alignment</td>
<td>Defines the alignment of the column in a report. Valid values are L (left), C (center) and R (right).</td>
</tr>
</tbody>
</table>

Example

In the following example, when creating a Report against the DEPT table, the default column alignment on the DEPTNO column is set to Right justified.

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

35.22 UPD_REPORT_FORMAT_MASK Procedure

The UPD_REPORT_FORMAT_MASK procedure sets the report format mask user interface default. This user interface default is used by wizards when you select to create a report based upon the table and include the specified column. Report format mask is typically used to format numbers and dates.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_REPORT_FORMAT_MASK (  
    p_table_name            IN VARCHAR2,  
    p_column_name           IN VARCHAR2,  
    p_format_mask           IN VARCHAR2 DEFAULT NULL);

Parameters

Table 35-22  UPD_REPORT_FORMAT_MASK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_column_name</td>
<td>Column name</td>
</tr>
<tr>
<td>p_format_mask</td>
<td>Format mask to be associated with the column whenever it is included in a report</td>
</tr>
</tbody>
</table>
Example

In the following example, when creating a Report against the EMP table, the default format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

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

35.23 UPD_REPORT_REGION_TITLE Procedure

The UPD_REPORT_REGION_TITLE procedure sets the Report Region Title. User interface defaults are used in wizards when a report is created on a table.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_REPORT_REGION_TITLE (  
    p_table_name            IN VARCHAR2,  
    p_report_region_title   IN VARCHAR2 DEFAULT NULL);

Parameters

Table 35-23  UPD_REPORT_REGION_TITLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Table name</td>
</tr>
<tr>
<td>p_report_region_title</td>
<td>Desired report region title</td>
</tr>
</tbody>
</table>

Example

This example demonstrates how to set the Reports Region Title user interface default on the DEPT table.

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

35.24 UPD_TABLE Procedure

If the provided table exists within the user's schema's table based User Interface Defaults, the form region title and report region title are updated to match those provided. If 'null%' is passed in for p_form_region_title or p_report_region_title, the value is set to null.

Syntax

APEX_UI_DEFAULT_UPDATE.UPD_TABLE (  
    p_table_name            IN VARCHAR2,
p_form_region_title IN VARCHAR2 DEFAULT NULL,
p_report_region_title IN VARCHAR2 DEFAULT NULL);

Parameters

Table 35-24  UPD_TABLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_table_name</td>
<td>Name of table being updated.</td>
</tr>
<tr>
<td>p_form_region_title</td>
<td>Region title used for forms.</td>
</tr>
<tr>
<td>p_report_region_title</td>
<td>Region title used for reports and tabular forms.</td>
</tr>
</tbody>
</table>

Note:
if 'null%' is passed in for p_form_region_title or p_report_region_title, the value is set to null. If no value is passed in, that column is not updated.

Example

The following example updates the EMP table definition within the UI Defaults Table Dictionary within the current schema.

begin
  apex_ui_default_update.upd_table (  
    p_table_name => 'EMP',
    p_form_region_title => 'Employee Details',
    p_report_region_title => 'Employees' );
end;
/

Chapter 35
UPD_TABLE Procedure
The APEX_UTIL package provides utilities you can use when programming in the Oracle Application Express environment. You can use the APEX_UTIL package to get and set session state, get files, check authorizations for users, reset different states for users, get and purge cache information and also to get and set preferences for users.

- CACHE_GET_DATE_OF_PAGE_CACHE Function
- CACHE_GET_DATE_OF_REGION_CACHE Function
- CACHE_PURGE_BY_APPLICATION Procedure
- CACHE_PURGE_BY_PAGE Procedure
- CACHE_PURGE_STALE Procedure
- CHANGE_CURRENT_USER_PW Procedure
- CHANGE_PASSWORD_ON_FIRST_USE Function
- CLOSE_OPEN_DB_LINKS Procedure
- CLEAR_APP_CACHE Procedure
- CLEAR_PAGE_CACHE Procedure
- CLEAR_USER_CACHE Procedure
- COUNT_CLICK Procedure
- CREATE_USER Procedure
- CREATE_USER_GROUP Procedure
- CURRENT_USER_IN_GROUP Function
- CUSTOM_CALENDAR Procedure
- DELETE_USER_GROUP Procedure Signature 1
- DELETE_USER_GROUP Procedure Signature 2
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4
- EDIT_USER Procedure
- END_USER_ACCOUNT_DAYS_LEFT Function
- EXPIRE_END_USER_ACCOUNT Procedure
- EXPIRE_WORKSPACE_ACCOUNT Procedure
- EXPORT_USERS Procedure
- FETCH_APP_ITEM Function
- FETCH_USER Procedure Signature 1
• FETCH_USER Procedure Signature 2
• FETCH_USER Procedure Signature 3
• FIND_SECURITY_GROUP_ID Function
• FIND_WORKSPACE Function
• GET_ACCOUNT_LOCKED_STATUS Function
• GET_APPLICATION_STATUS Function
• GET_ATTRIBUTE Function
• GET_AUTHENTICATION_RESULT Function
• GET_BLOB_FILE_SRC Function
• GET_BUILD_OPTION_STATUS Function Signature 1
• GET_BUILD_OPTION_STATUS Function Signature 2
• GET_CURRENT_USER_ID Function
• GET_DEFAULT_SCHEMA Function
• GET_EDITION Function
• GET_EMAIL Function
• GET_FEEDBACK_FOLLOW_UP Function
• GET_FILE Procedure
• GET_FILE_ID Function
• GET_FIRST_NAME Function
• GET_GLOBAL_NOTIFICATION Function
• GET_GROUPS_USER_BELongs_TO Function
• GET_GROUP_ID Function
• GET_GROUP_NAME Function
• GET_HASH Function
• GET_HIGH_CONTRAST_MODE_TOGGLE Function
• GET_LAST_NAME Function
• GET_NUMERIC_SESSION_STATE Function
• GET_PREFERENCE Function
• GET_PRINT_DOCUMENT Function Signature 1
• GET_PRINT_DOCUMENT Function Signature 2
• GET_PRINT_DOCUMENT Function Signature 3
• GET_PRINT_DOCUMENT Function Signature 4
• GET_SCREEN_READER_MODE_TOGGLE Function
• GET_SESSION_LANG Function
• GET_SESSION_STATE Function
• GET_SESSION_TERRITORY Function
• GET_SESSION_TIME_ZONE Function
• GET_SINCE Function
• GET_SUPPORTING_OBJECT_SCRIPT Function
• GET_SUPPORTING_OBJECT_SCRIPT Procedure
• GET_USER_ID Function
• GET_USER_ROLES Function
• GET_USERNAME Function
• HOST_URL Function
• HTML_PCT_GRAPH_MASK Function
• INCREMENTCALENDAR Procedure
• IR_CLEAR Procedure [DEPRECATED]
• IR_DELETE_REPORT Procedure [DEPRECATED]
• IR_DELETE_SUBSCRIPTION Procedure [DEPRECATED]
• IR_FILTER Procedure [DEPRECATED]
• IR_RESET Procedure [DEPRECATED]
• IS_HIGH_CONTRAST_SESSION Function
• IS_HIGH_CONTRAST_SESSION_YN Function
• IS_LOGIN_PASSWORD_VALID Function
• IS_SCREEN_READER_SESSION Function
• IS_SCREEN_READER_SESSION_YN Function
• IS_USERNAME_UNIQUE Function
• KEYVAL_NUM Function
• KEYVAL_VC2 Function
• LOCK_ACCOUNT Procedure
• PASSWORD_FIRST_USE_OCCURRED Function
• PREPARE_URL Function
• PUBLIC_CHECK_AUTHORIZATION Function [DEPRECATED]
• PURGE_REGIONS_BY_APP Procedure
• PURGE_REGIONS_BY_NAME Procedure
• PURGE_REGIONS_BY_PAGE Procedure
• REDIRECT_URL Procedure
• REMOVE_PREFERENCE Procedure
• REMOVE_SORT_PREFERENCES Procedure
• REMOVE_USER Procedure
• RESET_AUTHORIZATIONS Procedure [DEPRECATED]
• RESET_PASSWORD Procedure
• RESET_PW Procedure
• SAVEKEY_NUM Function
• SAVEKEY_VC2 Function
• SET_APP_BUILD_STATUS Procedure
• SET_APPLICATION_STATUS Procedure
• SET_ATTRIBUTE Procedure
• SET_AUTHENTICATION_RESULT Procedure
• SET_BUILD_OPTION_STATUS Procedure
• SET_CURRENT_THEME_STYLE Procedure [DEPRECATED]
• SET_CUSTOM_AUTH_STATUS Procedure
• SET_EDITION Procedure
• SET_EMAIL Procedure
• SET_FIRST_NAME Procedure
• SET_GLOBAL_NOTIFICATION Procedure
• SET_GROUP_GROUP_GRANTS Procedure
• SET_GROUP_USER_GRANTS Procedure
• SET_LAST_NAME Procedure
• SET_PARSING_SCHEMA_FOR_REQUEST Procedure
• SET_PREFERENCE Procedure
• SET_SECURITY_GROUP_ID Procedure
• SET_SESSION_HIGH_CONTRAST_OFF Procedure
• SET_SESSION_HIGH_CONTRAST_ON Procedure
• SET_SESSION_LANG Procedure
• SET_SESSION_LIFETIME_SECONDS Procedure
• SET_SESSION_MAX_IDLE_SECONDS Procedure
• SET_SESSION_SCREEN_READER_OFF Procedure
• SET_SESSION_SCREEN_READER_ON Procedure
• SET_SESSION_STATE Procedure
• SET_SESSION_TERRITORY Procedure
• SET_SESSION_TIME_ZONE Procedure
• SET_USERNAME Procedure
• SET_WORKSPACE Procedure
• SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure
• SHOW_SCREEN_READER_MODE_TOGGLE Procedure
• STRING_TO_TABLE Function [DEPRECATED]
• STRONG_PASSWORD_CHECK Procedure
• STRONG_PASSWORD_VALIDATION Function
• SUBMIT_FEEDBACK Procedure
• SUBMIT_FEEDBACK_FOLLOWUP Procedure
36.1 CACHE_GET_DATE_OF_PAGE_CACHE Function

This function returns the date and time a specified application page was cached either for the user issuing the call, or for all users if the page was not set to be cached by user.

Syntax

APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE (
   p_application  IN NUMBER,
   p_page         IN NUMBER)
RETURN DATE;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
<tr>
<td>p_page</td>
<td>The page number (ID).</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_PAGE_CACHE function to retrieve the cache date and time for page 9 of the currently executing application. If page 9 has been cached, the cache date and time is output using the HTP package. The page could have been cached either by the user issuing the call, or for all users if the page was not to be cached by the user.

DECLARE
   l_cache_date DATE DEFAULT NULL;
BEGIN
   l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE(
      p_application => :APP_ID,
      p_page => 9);
   IF l_cache_date IS NOT NULL THEN
      HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
   END IF;
END;
36.2 CACHE_GET_DATE_OF_REGION_CACHE Function

This function returns the date and time a specified region was cached either for the user issuing the call, or for all users if the page was not set to be cached by user.

Syntax

APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE (  
    p_application  IN NUMBER,  
    p_page         IN NUMBER,  
    p_region_name  IN VARCHAR2)  
RETURN DATE;

Parameters

Table 36-2  CACHE_GET_DATE_OF_REGION_CACHE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
<tr>
<td>p_page</td>
<td>The page number (ID).</td>
</tr>
<tr>
<td>p_region_name</td>
<td>The region name.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_REGION_CACHE function to retrieve the cache date and time for the region named Cached Region on page 13 of the currently executing application. If the region has been cached, the cache date and time is output using the HTP package. The region could have been cached either by the user issuing the call, or for all users if the page was not to be cached by user.

DECLARE  
    l_cache_date DATE DEFAULT NULL;  
BEGIN  
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE(  
        p_application => :APP_ID,  
        p_page => 13,  
        p_region_name => 'Cached Region');  
    IF l_cache_date IS NOT NULL THEN  
        HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));  
    END IF;  
END;

36.3 CACHE_PURGE_BY_APPLICATION Procedure

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

APEX_UTIL.CACHE_PURGE_BY_APPLICATION (p_application IN NUMBER);

Parameters

Table 36-3  CACHE_PURGE_BY_APPLICATION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CACHE_PURGE_BY_APPLICATION procedure to purge all the cached pages and regions for the application currently executing.

BEGIN
  APEX_UTIL.CACHE_PURGE_BY_APPLICATION(p_application => :APP_ID);
END;

36.4 CACHE_PURGE_BY_PAGE Procedure

This procedure purges the cache for a given application and page. If the page itself is not cached but contains one or more cached regions, then the cache for these is also purged.

Syntax

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

Parameters

Table 36-4  CACHE_PURGE_BY_PAGE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
<tr>
<td>p_page</td>
<td>The page number (ID).</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The user associated with cached pages and regions.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CACHE_PURGE_BY_PAGE procedure to purge the cache for page 9 of the application currently executing. Additionally, if the
p_user_name parameter is supplied, this procedure would be further restricted by a specific users cache (only relevant if the cache is set to be by user).

BEGIN
    APEX_UTIL.CACHE_PURGE_BY_PAGE(
        p_application => :APP_ID,
        p_page => 9);
END;

36.5 CACHE_PURGE_STALE Procedure

This procedure deletes all cached pages and regions for a specified application that have passed the defined active time period. When you cache a page or region, you specify an active time period (or Cache Timeout). Once that period has passed, the cache is no longer used, thus removing those unusable pages or regions from the cache.

Syntax

APEX_UTIL.CACHE_PURGE_STALE (  
    p_application  IN NUMBER);

Parameters

Table 36-5  CACHE_PURGE_STALE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CACHE_PURGE_STALE procedure to purge all the stale pages and regions in the application currently executing.

BEGIN
    APEX_UTIL.CACHE_PURGE_STALE(p_application => :APP_ID);
END;

36.6 CHANGE_CURRENT_USER_PW Procedure

This procedure changes the password of the currently authenticated user, assuming Application Express user accounts are in use.

Syntax

APEX_UTIL.CHANGE_CURRENT_USER_PW(  
    p_new_password IN VARCHAR2);
Parameters

Table 36-6  CHANGE_CURRENT_USER_PW Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_new_password</td>
<td>The new password value in clear text.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CHANGE_CURRENT_USER_PW procedure to change the password for the user who is currently authenticated, assuming Application Express accounts are in use.

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

See Also:

"RESET_PW Procedure"

36.7 CHANGE_PASSWORD_ON_FIRST_USE Function

Enables a developer to check whether this property is enabled or disabled for an end user account. This function returns TRUE if the account password must be changed upon first use (after successful authentication) after the password is initially set and after it is changed on the Administration Service, Edit User page. This function returns FALSE if the account does not have this property.

This function may be run in a page request context by any authenticated user.

Syntax

APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE ( p_user_name IN VARCHAR2)
RETURN BOOLEAN;

Parameters

Table 36-7  CHANGE_PASSWORD_ON_FIRST_USE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>
Example

The following example demonstrates how to use the `CHANGE_PASSWORD_ON_FIRST_USE` function. Use this function to check if the password of an Application Express user account (workspace administrator, developer, or end user) in the current workspace must be changed by the user the first time it is used.

```sql
BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) LOOP
        IF APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE(p_user_name => c1.user_name) THEN
            htp.p('User: ' || c1.user_name || ' requires password to be changed the first time it is used.');
        END IF;
    END LOOP;
END;
```

See Also:

"PASSWORD_FIRST_USE_OCCURRED Function"

36.8 CLOSE_OPEN_DB_LINKS Procedure

This procedure closes all open database links for the current database session. It is rare that this procedure would ever be called programmatically in an application. The primary purpose of this procedure is for the middleware technology in an Oracle Application Express environment (for example, Oracle REST Data Service, mod_plsql) to be configured such that it closes all of the open database links in a session, either before a request is made to the Application Express engine, or after a request to the Application Express engine is completed but before the database session is returned to the pool.

Syntax

APEX_UTIL.CLOSE_OPEN_DB_LINKS

Parameters

None.

Example

In this example, the configuration of Oracle REST Data Services closes any open database links both before the request is made to the Application Express engine and after the request is complete.

```xml
<entry key="procedure.postProcess">apex_util.close_open_db_links</entry>
<entry key="procedure.preProcess">apex_util.close_open_db_links</entry>
```
When using Oracle HTTP Server and mod_plsql, this configuration would look like this:

```
PlsqlBeforeProcedure    apex_util.close_open_db_links
PlsqlAfterProcedure        apex_util.close_open_db_links
```

### 36.9 CLEAR_APP_CACHE Procedure

This procedure removes session state for a given application for the current session.

#### Syntax

```
APEX_UTIL.CLEAR_APP_CACHE (p_app_id IN VARCHAR2 DEFAULT NULL);
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_app_id</td>
<td>The ID of the application for which session state is cleared for current session.</td>
</tr>
</tbody>
</table>

#### Example

The following example demonstrates how to use the CLEAR_APP_CACHE procedure to clear all the current sessions state for the application with an ID of 100.

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

### 36.10 CLEAR_PAGE_CACHE Procedure

This procedure removes session state for a given page for the current session. If `p_page_id` is not specified, then the current page will be cleared.

#### Syntax

```
APEX_UTIL.CLEAR_PAGE_CACHE ( p_page_id IN NUMBER DEFAULT NULL);  
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>The ID of the page in the current application for which session state is cleared for current session.</td>
</tr>
</tbody>
</table>
Example

The following example demonstrates how to use the CLEAR_PAGE_CACHE procedure to clear the current session state for the page with an ID of 10.

BEGIN
    APEX_UTIL.CLEAR_PAGE_CACHE(10);
END;

36.11 CLEAR_USER_CACHE Procedure

This procedure removes session state and application system preferences for the current user's session. Run this procedure if you reuse session IDs and want to run applications without the benefit of existing session state.

Syntax

APEX_UTIL.CLEAR_USER_CACHE;

Parameters

None.

Example

The following example demonstrates how to use the CLEAR_USER_CACHE procedure to clear all session state and application system preferences for the current user's session.

BEGIN
    APEX_UTIL.CLEAR_USER_CACHE;
END;

36.12 COUNT_CLICK Procedure

This procedure counts clicks from an application built in App Builder to an external site. You can also use the shorthand version, procedure Z_COUNT_CLICK, 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 36-10 COUNT_CLICK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The URL to which to redirect</td>
</tr>
<tr>
<td>p_cat</td>
<td>A category to classify the click</td>
</tr>
<tr>
<td>p_id</td>
<td>Secondary ID to associate with the click (optional)</td>
</tr>
<tr>
<td>p_user</td>
<td>The application user ID (optional)</td>
</tr>
<tr>
<td>p_workspace</td>
<td>The workspace associated with the application (optional)</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the COUNT_CLICK procedure to log how many user's click on the http://yahoo.com link specified. Note that once this information is logged, you can view it by using the APEX_WORKSPACE_CLICKS view and in the reports on this view available to workspace and site administrators.

```sql
DECLARE
  l_url VARCHAR2(255);
  l_cat VARCHAR2(30);
  l_workspace_id VARCHAR2(30);
BEGIN
  l_url := 'http://yahoo.com';
  l_cat := 'yahoo';
  l_workspace_id := TO_CHAR(APEX_UTIL.FIND_SECURITY_GROUP_ID('MY_WORKSPACE'));
  HTP.P('<a href=APEX_UTIL.COUNT_CLICK?p_url=' || l_url || '&p_cat=' || l_cat || '&p_workspace=' || l_workspace_id || '>Click</a>');
END;
```

See Also:

- "FIND_SECURITY_GROUP_ID Function" in this document and "Deleting Click Counting Log Entries" in Oracle Application Express Administration Guide
- "Managing Authorized URLs" in Oracle Application Express Administration Guide

36.13 CREATE_USER Procedure

This procedure creates a new account record in the Application Express user account table. To execute this procedure, the current user must have administrative privileges.
Syntax

APEX_UTIL.CREATE_USER(
    p_user_id                       IN      NUMBER      DEFAULT NULL,
    p_user_name                     IN      VARCHAR2,
    p_first_name                    IN      VARCHAR2    DEFAULT NULL,
    p_last_name                     IN      VARCHAR2    DEFAULT NULL,
    p_description                   IN      VARCHAR2    DEFAULT NULL,
    p_email_address                 IN      VARCHAR2    DEFAULT NULL,
    p_web_password                  IN      VARCHAR2,
    p_web_password_format           IN      VARCHAR2    DEFAULT 'CLEAR_TEXT',
    'CLEAR_TEXT',
    p_group_ids                     IN      VARCHAR2    DEFAULT NULL,
    p_developer_privs               IN      VARCHAR2    DEFAULT NULL,
    p_default_schema                IN      VARCHAR2    DEFAULT NULL,
    p_account_expiry                IN      DATE        DEFAULT TRUNC(SYSDATE),
    p_account_locked                IN      VARCHAR2    DEFAULT 'N',
    p_failed_access_attempts       IN      NUMBER      DEFAULT 0,
    p_change_password_on_first_use  IN      VARCHAR2    DEFAULT 'N',
    p_first_password_use_occurred   IN      VARCHAR2    DEFAULT 'N',
    p_attribute_01                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_02                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_03                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_04                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_05                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_06                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_07                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_08                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_09                  IN      VARCHAR2    DEFAULT NULL,
    p_attribute_10                  IN      VARCHAR2    DEFAULT NULL,
    p_allow_app_building_yn         IN      VARCHAR2    DEFAULT NULL,
    p_allow_sql_workshop_yn         IN      VARCHAR2    DEFAULT NULL,
    p_allow_websheet_dev_yn         IN      VARCHAR2    DEFAULT NULL,
    p_allow_team_development_yn     IN      VARCHAR2    DEFAULT NULL);

Parameters

Table 36-11  CREATE_USER Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_id</td>
<td>Numeric primary key of user account.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>Alphanumeric name used for login.</td>
</tr>
<tr>
<td>p_first_name</td>
<td>Informational.</td>
</tr>
<tr>
<td>p_last_name</td>
<td>Informational.</td>
</tr>
<tr>
<td>p_description</td>
<td>Informational.</td>
</tr>
<tr>
<td>p_email_address</td>
<td>Email address.</td>
</tr>
<tr>
<td>p_web_password</td>
<td>Clear text password.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>p_web_password_format</td>
<td>If the value your passing for the <code>p_web_password</code> parameter is in clear text format then use <code>CLEAR_TEXT</code>, otherwise use <code>HEX_ENCODED_DIGEST_V2</code>.</td>
</tr>
<tr>
<td>p_group_ids</td>
<td>Colon separated list of numeric group IDs.</td>
</tr>
<tr>
<td>p_developer_prives</td>
<td>Colon separated list of developer privileges. If <code>p_developer_prives</code> is not null, the user is given access to Team Development. If <code>p_developer_prives</code> contains ADMIN, the user is given App Builder and SQL Workshop access. If <code>p_developer_prives</code> does not contain ADMIN but contains EDIT, the user is given App Builder Access. If <code>p_developer_prives</code> does not contain ADMIN but contains SQL, the user is given SQL Workshop access. The following are acceptable values for this parameter:</td>
</tr>
<tr>
<td></td>
<td><code>null</code> - To create an end user (a user who can only authenticate to developed applications).</td>
</tr>
<tr>
<td></td>
<td><code>CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL</code> - To create a user with developer privileges with access to App Builder and SQL Workshop.</td>
</tr>
<tr>
<td></td>
<td><code>ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL</code> - To create a user with full workspace administrator and developer privileges with access to App Builder, SQL Workshop and Team Development. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses <code>p_developer_prives</code>, EDIT_USER uses <code>p_developer_roles</code> and FETCH_USER uses <code>p_developer_role</code>.</td>
</tr>
<tr>
<td>p_default_schema</td>
<td>A database schema assigned to the user's workspace, used by default for browsing.</td>
</tr>
<tr>
<td>p_allow_access_to_schemas</td>
<td>Colon separated list of schemas assigned to the user's workspace to which the user is restricted (leave null for all).</td>
</tr>
<tr>
<td>p_account_expiry</td>
<td>Date password was last updated, which defaults to today's date on creation.</td>
</tr>
<tr>
<td>p_account_locked</td>
<td>'Y' or 'N' indicating if account is locked or unlocked.</td>
</tr>
<tr>
<td>p_failed_access_attempts</td>
<td>Number of consecutive login failures that have occurred, defaults to 0 on creation.</td>
</tr>
<tr>
<td>p_change_password_on_first_use</td>
<td>'Y' or 'N' to indicate whether password must be changed on first use, defaults to 'Y' on creation.</td>
</tr>
<tr>
<td>p_first_password_useOccurred</td>
<td>'Y' or 'N' to indicate whether login has occurred since password change, defaults to 'N' on creation.</td>
</tr>
</tbody>
</table>
Table 36-11  (Cont.) CREATE_USER Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_attribute_01</td>
<td>Arbitrary text accessible with an API.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>p_attribute_10</td>
<td></td>
</tr>
<tr>
<td>p_allow_app_building_yn</td>
<td>'Y' or 'N' to indicate whether access is allowed to App Builder.</td>
</tr>
<tr>
<td>p_allow_sql_workshop_yn</td>
<td>'Y' or 'N' to indicate whether access is allowed to SQL Workshop.</td>
</tr>
<tr>
<td>p_allow_websheet_dev_yn</td>
<td>'Y' or 'N' to indicate whether access is allowed to Websheet development.</td>
</tr>
<tr>
<td>p_allow_team_development_yn</td>
<td>'Y' or 'N' to indicate whether access is allowed to Team Development.</td>
</tr>
</tbody>
</table>

Example 1

The following simple example creates an 'End User' called 'NEWUSER1' with a password of 'secret99'. Note an 'End User' can only authenticate to developed applications.

BEGIN
APEX_UTIL.CREATE_USER(
    p_user_name    => 'NEWUSER1',
    p_web_password => 'secret99');
END;

Example 2

The following example creates a 'Workspace Administrator' called 'NEWUSER2'. Where the user 'NEWUSER2':

- Has full workspace administration and developer privilege (p_developer_privs parameter set to 'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL').
- Has access to 2 schemas, both their browsing default 'MY_SCHEMA' (p_default_schema parameter set to 'MY_SCHEMA') and also 'MY_SCHEMA2' (p_allow_access_to_schemas parameter set to 'MY_SCHEMA2').
- Does not have to change their password when they first login (p_change_password_on_first_use parameter set to 'N').
- Has their phone number stored in the first additional attribute (p_attribute_01 parameter set to '123 456 7890').

BEGIN
APEX_UTIL.CREATE_USER(
    p_user_name                     => 'NEWUSER2',
    p_first_name                    => 'FRANK',
    p_last_name                     => 'SMITH',
    p_description                   => 'Description...',
    p_email_address                 => 'frank@smith.com',
    p_web_password                  => 'password',
    p_attribute_01                  => '123 456 7890');
END;
36.14 CREATE_USER_GROUP Procedure

Assuming you are using Application Express authentication, this procedure creates a user group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

APEX_UTIL.CREATE_USER_GROUP(
  p_id                      IN NUMBER default null,
  p_group_name              IN ARCHAR2,
  p_security_group_id       IN NUMBER default null,
  p_group_desc              IN VARCHAR2 default null,);

Parameter

Table 36-12 CREATE_USER_GROUP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_id</td>
<td>Primary key of group.</td>
</tr>
<tr>
<td>p_group_name</td>
<td>Name of group.</td>
</tr>
<tr>
<td>p_security_group_id</td>
<td>Workspace ID.</td>
</tr>
<tr>
<td>p_group_desc</td>
<td>Descriptive text.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CREATE_USER_GROUP procedure to create a new group called 'Managers' with a description of 'text'. Pass null for the p_id.

```sql
APEX_UTIL.CREATE_USER_GROUP(
  p_group_name => 'Managers',
  p_group_desc => 'text');
```
parameter to allow the database trigger to assign the new primary key value. Pass null for the p_security_group_id parameter to default to the current workspace ID.

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

36.15 CURRENT_USER_IN_GROUP Function

This function returns a Boolean result based on whether the current user is a member of the specified group. You can use the group name or group ID to identify the group.

Syntax

APEX_UTIL.CURRENT_USER_IN_GROUP(
   p_group_name  IN VARCHAR2)
RETURN BOOLEAN;

APEX_UTIL.CURRENT_USER_IN_GROUP(
   p_group_id    IN NUMBER)
RETURN BOOLEAN;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_name</td>
<td>Identifies the name of an existing group in the workspace</td>
</tr>
<tr>
<td>p_group_id</td>
<td>Identifies the numeric ID of an existing group in the workspace</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the CURRENT_USER_IN_GROUP function to check if the user currently authenticated belongs to the group 'Managers'.

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

36.16 CUSTOM_CALENDAR Procedure

Use this procedure to change the existing calendar view to Custom Calendar.
Syntax

APEX_UTIL.CUSTOM_CALENDAR(
    p_date_type_field IN VARCHAR2);

Parameters

Table 36-14   CUSTOM_CALENDAR Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_date_type_field</td>
<td>Identifies the item name used to define the type of calendar to be displayed.</td>
</tr>
</tbody>
</table>

Example 1

The following example defines a custom calendar based on the hidden calendar type field. Assuming the Calendar is created in Page 9, the following example hides the column called P9_CALENDAR_TYPE.

APEX_UTIL.CUSTOM_CALENDAR(
    'P9_CALENDAR_TYPE');

36.17 DELETE_USER_GROUP Procedure Signature 1

Assuming you are using Application Express authentication, this procedure deletes a user group by providing the primary key of the group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

APEX_UTIL.DELETE_USER_GROUP(
    p_group_id IN NUMBER);

Parameter

Table 36-15   DELETE_USER_GROUP Procedure Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_id</td>
<td>Primary key of group.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the DELETE_USER_GROUP procedure signature 1 to remove the user group called 'Managers', by providing the user group's primary key.

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

36.18 DELETE_USER_GROUP Procedure Signature 2

Assuming you are using Application Express authentication, this procedure deletes a user group by providing the name of the group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

APEX_UTIL.DELETE_USER_GROUP (p_group_name IN VARCHAR2);

Parameter

Table 36-16  DELETE_USER_GROUP Procedure Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_name</td>
<td>Name of group</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the DELETE_USER_GROUP procedure signature 2 to remove the user group called 'Managers', by providing the name of the user group.

BEGIN
    APEX_UTIL.DELETE_USER_GROUP (p_group_name => 'Managers');
END;

36.19 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1

This procedure initiates the download of a print document using XML based report data (as a BLOB) and RTF or XSL-FO based report layout.

Syntax

APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (p_file_name IN VARCHAR,
p_content_disposition IN VARCHAR,
p_report_data IN BLOB,
p_report_layout IN CLOB,
p_report_layout_type IN VARCHAR2 default 'xsl-fo',)
DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2

This procedure initiates the download of a print document using pre-defined report query and RTF and XSL-FO based report layout.

Syntax

APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (  
    p_file_name           IN VARCHAR,  
    p_content_disposition IN VARCHAR,  
    p_application_id      IN NUMBER,  
    p_report_query_name   IN VARCHAR2,  
    p_report_layout       IN CLOB,  
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',  
    p_document_format     IN VARCHAR2 default 'pdf',  
    p_print_server        IN VARCHAR2 default null);
Parameters

Table 36-18  DOWNLOAD_PRINT_DOCUMENT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_file_name</td>
<td>Defines the filename of the print document.</td>
</tr>
<tr>
<td>p_content_disposition</td>
<td>Specifies whether to download the print document or display inline (&quot;attachment&quot;, &quot;inline&quot;).</td>
</tr>
<tr>
<td>p_application_id</td>
<td>Defines the application ID of the report query.</td>
</tr>
<tr>
<td>p_report_query_name</td>
<td>Name of the report query (stored under application's Shared Components).</td>
</tr>
<tr>
<td>p_report_layout</td>
<td>Report layout in XSL-FO or RTF format.</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

Example for Signature 2

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 2 (Pre-defined report query and RTF or XSL-FO based report layout.). In this example, the data for the report is taken from a Report Query called 'ReportQueryAndXSL' stored in the current application's Shared Components > Report Queries. The report layout is taken from a value stored in a page item (P1_XSL).

BEGIN
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name => 'mydocument',
    p_content_disposition => 'attachment',
    p_application_id => :APP_ID,
    p_report_query_name => 'ReportQueryAndXSL',
    p_report_layout => :P1_XSL,
    p_report_layout_type => 'xsl-fo',
    p_document_format => 'pdf');
END;

See Also:

36.21 DOWNLOAD_PRINT_DOCUMENT Procedure

Signature 3

This procedure initiates the download of a print document using pre-defined report query and pre-defined report layout.

Syntax

APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (  
    p_file_name       IN VARCHAR,  
    p_content_disposition IN VARCHAR,  
    p_application_id   IN NUMBER,  
    p_report_query_name IN VARCHAR2,  
    p_report_layout_name  IN VARCHAR2,  
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',  
    p_document_format    IN VARCHAR2 default 'pdf',  
    p_print_server      IN VARCHAR2 default null);

Parameters

Table 36-19  DOWNLOAD_PRINT_DOCUMENT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_file_name</td>
<td>Defines the filename of the print document.</td>
</tr>
<tr>
<td>p_content_disposition</td>
<td>Specifies whether to download the print document or display inline (&quot;attachment&quot;, &quot;inline&quot;).</td>
</tr>
<tr>
<td>p_application_id</td>
<td>Defines the application ID of the report query.</td>
</tr>
<tr>
<td>p_report_query_name</td>
<td>Name of the report query (stored under application's Shared Components).</td>
</tr>
<tr>
<td>p_report_layout_name</td>
<td>Name of the report layout (stored under application's Shared Components).</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

Example for Signature 3

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 3 (Pre-defined report query and pre-defined report layout). In this example, the data for the report is taken from a Report Query called 'ReportQuery' stored in the current application's Shared Components > Report Queries. The report layout is taken from a Report Layout called 'ReportLayout' stored in the current application's Shared Components > Report Layouts. Note that if you want to provision dynamic layouts, instead of specifying 'ReportLayout' for the p_report_layout_name parameter, you could reference a page item that allowed the user to select one of multiple saved Report Layouts. This example also provides a way for the user to specify how they
want to receive the document (as an attachment or inline), through passing the value of P1_CONTENT_DISP to the p_content_disposition parameter. P1_CONTENT_DISP is a page item of type 'Select List' with the following List of Values Definition:

STATIC2:In Browser;inline,Save / Open in separate Window;attachment

BEGIN
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (  
    p_file_name           => 'myreport123',  
    p_content_disposition => P1_CONTENT_DISP,  
    p_application_id      => APP_ID,  
    p_report_query_name   => 'ReportQuery',  
    p_report_layout_name  => 'ReportLayout',  
    p_report_layout_type  => 'rtf',  
    p_document_format     => 'pdf');
END;

See Also:

36.22 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4

This procedure initiates the download of a print document using XML based report data (as a CLOB) and RTF or XSL-FO based report layout.

Syntax

APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (  
    p_file_name           IN VARCHAR,  
    p_content_disposition IN VARCHAR,  
    p_report_data         IN CLOB,  
    p_report_layout       IN CLOB,  
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',  
    p_document_format     IN VARCHAR2 default 'pdf',  
    p_print_server        IN VARCHAR2 default null);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_file_name</td>
<td>Defines the filename of the print document.</td>
</tr>
<tr>
<td>p_content_disposition</td>
<td>Specifies whether to download the print document or display inline (&quot;attachment&quot;, &quot;inline&quot;).</td>
</tr>
</tbody>
</table>
Table 36-20  (Cont.) DOWNLOAD_PRINT_DOCUMENT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_report_data</td>
<td>XML based report data, must be encoded in UTF-8.</td>
</tr>
<tr>
<td>p_report_layout</td>
<td>Report layout in XSL-FO or RTF format.</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

Example for Signature 4

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 4 (XML based report data (as a CLOB) and RTF or XSL-FO based report layout). In this example both the report data (XML) and report layout (XSL-FO) are taken from values stored in page items.

BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (  
        p_file_name        => 'mydocument',  
        p_content_disposition => 'attachment',  
        p_report_data      => :P1_XML,  
        p_report_layout    => :P1_XSL,  
        p_report_layout_type => 'xsl-fo',  
        p_document_format  => 'pdf');
END;

See Also:


36.23 EDIT_USER Procedure

This procedure enables a user account record to be altered. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

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

Parameters

Table 36-21    EDIT_USER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_id</td>
<td>Numeric primary key of the user account.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>Alphanumeric name used for login.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;SET_USERNAME Procedure&quot;</td>
</tr>
<tr>
<td>p_first_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;SET_FIRST_NAME Procedure&quot;</td>
</tr>
<tr>
<td>p_last_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;SET_LAST_NAME Procedure&quot;</td>
</tr>
<tr>
<td>p_web_password</td>
<td>Clear text password. If using this procedure to update the password for the</td>
</tr>
<tr>
<td></td>
<td>user, values for both p_web_password and p_new_password must not be null</td>
</tr>
<tr>
<td></td>
<td>and must be identical.</td>
</tr>
</tbody>
</table>
### Table 36-21  (Cont.) EDIT_USER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_new_password</td>
<td>Clear text new password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.</td>
</tr>
<tr>
<td>p_email_address</td>
<td>Informational.</td>
</tr>
<tr>
<td>p_start_date</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_end_date</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_employee_id</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_allow_access_to_schemas</td>
<td>A list of schemas assigned to the user’s workspace to which the user is restricted.</td>
</tr>
<tr>
<td>p_person_type</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_default_schema</td>
<td>A database schema assigned to the user’s workspace, used by default for browsing.</td>
</tr>
<tr>
<td>p_group_ids</td>
<td>Colon-separated list of numeric group IDs.</td>
</tr>
</tbody>
</table>
| p_developer_roles                  | Colon-separated list of developer privileges. The following are acceptable values for this parameter:  
  · null - To update the user to be an end user (a user who can only authenticate to developed applications).  
  · CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - To update the user to have developer privilege.    
  · ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - To update the user to have full workspace administrator and developer privilege.  
  **Note:** Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.  
  **See Also:** "GET_USER_ROLES Function"                                                                                                             |
| 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.                                                                                       |
Table 36-21  (Cont.) EDIT_USER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_change_password_on_first_use</td>
<td>'Y' or 'N' to indicate whether password must be changed on first use.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;CHANGE_PASSWORD_ON_FIRST_USE Function&quot;</td>
</tr>
<tr>
<td>p_first_password_useOccurred</td>
<td>'Y' or 'N' to indicate whether login has occurred since password change.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;PASSWORD_FIRST_USE_OCCURRED Function&quot;</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the EDIT_USER procedure to update a user account. This example shows how you can use the EDIT_USER procedure to change the user ‘FRANK’ from a user with just developer privilege to a user with workspace administrator and developer privilege. Firstly, the FETCH_USER procedure is called to assign account details for the user ‘FRANK’ to local variables. These variables are then used in the call to EDIT_USER to preserve the details of the account, with the exception of the value for the p_developer_roles parameter, which is set to 'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL'.

DECLARE
    l_user_id                       NUMBER;
    l_workspace                     VARCHAR2(255);
    l_user_name                     VARCHAR2(100);
    l_first_name                    VARCHAR2(255);
    l_last_name                     VARCHAR2(255);
    l_web_password                  VARCHAR2(255);
    l_email_address                 VARCHAR2(240);
    l_start_date                    DATE;
    l_end_date                      DATE;
    l_employee_id                   NUMBER(15,0);
    l_allow_access_to_schemas       VARCHAR2(4000);
    l_person_type                   VARCHAR2(1);
    l_default_schema                VARCHAR2(30);
    l_groups                        VARCHAR2(1000);
    l_developer_role                VARCHAR2(60);
    l_description                   VARCHAR2(240);
    l_account_expiry                DATE;
    l_account_locked                VARCHAR2(1);
    l_failed_access_attempts        NUMBER;
    l_change_password_on_first_use  VARCHAR2(1);
    l_first_password_use_occurred   VARCHAR2(1);
BEGIN
    l_user_id := APEX_UTIL.GET_USER_ID('FRANK');

    APEX_UTIL.FETCH_USER{
        p_user_id => l_user_id,
        p_workspace => l_workspace,
        p_user_name => l_user_name,
    }
```sql
APEX_UTIL.EDIT_USER (  
p_user_id                       => l_user_id,  
p_user_name                     => l_user_name,  
p_first_name                    => l_first_name,  
p_last_name                     => l_last_name,  
p_web_password                  => l_web_password,  
p_new_password                  => l_web_password,  
p_email_address                 => l_email_address,  
p_start_date                    => l_start_date,  
p_end_date                      => l_end_date,  
p_employee_id                   => l_employee_id,  
p_allow_access_to_schemas       => l_allow_access_to_schemas,  
p_person_type                   => l_person_type,  
p_default_schema                => l_default_schema,  
p_groups                        => l_groups,  
p_description                   => l_description,  
p_account_expiry                => l_account_expiry,  
p_account_locked                => l_account_locked,  
p_failed_access_attempts        => l_failed_access_attempts,  
p_change_password_on_first_use  => l_change_password_on_first_use,  
p_first_password_use_occurred   => l_first_password_use_occurred);

END;
```

See Also:

"FETCH_USER Procedure Signature 3"
36.24 END_USER_ACCOUNT_DAYS_LEFT Function

Returns the number of days remaining before an 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 36-22   END_USER_ACCOUNT_DAYS_LEFT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the END_USER_ACCOUNT_DAYS_LEFT function. Use this function to determine the number of days remaining before an Application Express end user account in the current workspace expires.

DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        l_days_left := APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT(p_user_name => c1.user_name);
        htp.p('End User Account:||c1.user_name||' expires in '||
             l_days_left||' days.');
    END LOOP;
END;

See Also:

- "EXPIRE_END_USER_ACCOUNT Procedure"
- "UNEXPIRE_END_USER_ACCOUNT Procedure"

36.25 EXPIRE_END_USER_ACCOUNT Procedure

Expires the login account for use as a workspace end user. Must be run by an authenticated workspace administrator in a page request context.
Syntax

APEX_UTIL.EXPIRE_END_USER_ACCOUNT (  
   p_user_name IN VARCHAR2  
);

Parameters

Table 36-23  EXPIRE_END_USER_ACCOUNT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the EXPIRE_END_USER_ACCOUNT procedure. Use this procedure to expire an Oracle Application Express account (workspace administrator, developer, or end user) in the current workspace. This action specifically expires the account for its use by end users to authenticate to developed applications, but it may also expire the account for its use by developers or administrators to log in to a workspace.

Note that this procedure must be run by a user having administration privileges in the current workspace.

BEGIN  
   FOR c1 IN (select user_name from apex_users) LOOP  
      APEX_UTIL.EXPIRE_END_USER_ACCOUNT(p_user_name => c1.user_name);  
      htp.p('End User Account:'||c1.user_name||' is now expired.');  
   END LOOP;  
END;

See Also:

"UNEXPIRE_END_USER_ACCOUNT Procedure"

36.26 EXPIRE_WORKSPACE_ACCOUNT Procedure

Expires developer or workspace administrator login accounts. Must be run by an authenticated workspace administrator in a page request context.

Syntax

APEX_UTIL.EXPIRE_WORKSPACE_ACCOUNT (  
   p_user_name IN VARCHAR2  
);
Parameters

Table 36-24  EXPIRE_WORKSPACE_ACCOUNT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the EXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to expire an Application Express account (workspace administrator, developer, or end user) in the current workspace. This action specifically expires the account for its use by developers or administrators to log in to a workspace, but it may also expire the account for its use by end users to authenticate to developed applications.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) LOOP
        APEX_UTIL.EXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account: ' || c1.user_name || ' is now expired.');
    END LOOP;
END;
```

See Also:

"UNEXPIRE_WORKSPACE_ACCOUNT Procedure"

36.27 EXPORT_USERS Procedure

When called from a page, this procedure produces an export file of the current workspace definition, workspace users, and workspace groups. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.EXPORT_USERS{
    p_export_format IN VARCHAR2 DEFAULT 'UNIX';
}
```
Parameters

Table 36-25  EXPORT_USERS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_export_format</td>
<td>Indicates how rows in the export file are formatted. Specify 'UNIX' to have the resulting file contain rows delimited by line feeds. Specify 'DOS' to have the resulting file contain rows delimited by carriage returns and line feeds.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the EXPORT_USERS procedure. Call this procedure from a page to produce an export file containing the current workspace definition, list of workspace users and list of workspace groups. The file is formatted with rows delimited by line feeds.

BEGIN
    APEX_UTIL.EXPORT_USERS;
END;

36.28 FETCH_APP_ITEM Function

This function fetches session state for the current or specified application in the current or specified session.

Syntax

APEX_UTIL.FETCH_APP_ITEM(
    p_item IN VARCHAR2,
    p_app IN NUMBER DEFAULT NULL,
    p_session IN NUMBER DEFAULT NULL)
RETURN VARCHAR2;

Parameters

Table 36-26  FETCH_APP_ITEM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item</td>
<td>The name of an application-level item (not a page item) whose current value is to be fetched.</td>
</tr>
<tr>
<td>p_app</td>
<td>The ID of the application that owns the item (leave null for the current application).</td>
</tr>
<tr>
<td>p_session</td>
<td>The session ID from which to obtain the value (leave null for the current session).</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the `FETCH_APP_ITEM` function to obtain the value of the application item 'F300_NAME' in application 300. As no value is passed for `p_session`, this defaults to the current session state value.

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

36.29 FETCH_USER Procedure Signature 1

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 1

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

Parameters for Signature 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_id</td>
<td>Numeric primary key of the user account.</td>
</tr>
<tr>
<td>p_workspace</td>
<td>The name of the workspace.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>Alphanumeric name used for login.</td>
</tr>
</tbody>
</table>

See Also: “GET_USERNAME Function”
## Table 36-27  (Cont.) Fetch_User Parameters Signature 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_first_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_FIRST_NAME Function&quot;</td>
</tr>
<tr>
<td>p_last_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_LAST_NAME Function&quot;</td>
</tr>
<tr>
<td>p_web_password</td>
<td>Obfuscated account password.</td>
</tr>
<tr>
<td>p_email_address</td>
<td>Email address.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_EMAIL Function&quot;</td>
</tr>
<tr>
<td>p_start_date</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_end_date</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_employee_id</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_allow_access_to_schemas</td>
<td>A list of schemas assigned to the user's workspace to which user is restricted.</td>
</tr>
<tr>
<td>p_person_type</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_default_schema</td>
<td>A database schema assigned to the user's workspace, used by default for browsing.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_DEFAULT_SCHEMA Function&quot;</td>
</tr>
<tr>
<td>p_groups</td>
<td>List of groups of which user is a member.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_GROUPS_USER_BELONGS_TO Function&quot; and &quot;CURRENT_USER_IN_GROUP Function&quot;</td>
</tr>
<tr>
<td>p_developer_role</td>
<td>Colon-separated list of developer roles. The following are acceptable values for this parameter:</td>
</tr>
<tr>
<td></td>
<td>null - Indicates an end user (a user who can only authenticate to developed applications).</td>
</tr>
<tr>
<td></td>
<td>Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_USER_ROLES Function&quot;</td>
</tr>
<tr>
<td>p_description</td>
<td>Informational.</td>
</tr>
</tbody>
</table>

## Example for Signature 1

The following example shows how to use the `FETCH_USER` procedure with Signature 1. This procedure is passed the ID of the currently authenticated user for the only...
parameter \texttt{p\_user\_id}. The code then stores all the other \texttt{OUT} parameter values in local variables.

\begin{verbatim}
DECLARE
  l_workspace     VARCHAR2(255);
  l_user_name     VARCHAR2(100);
  l_first_name    VARCHAR2(255);
  l_last_name     VARCHAR2(255);
  l_web_password  VARCHAR2(255);
  l_email_address VARCHAR2(240);
  l_start_date    DATE;
  l_end_date      DATE;
  l_employee_id   NUMBER(15,0);
  l_allow_access_to_schemas VARCHAR2(4000);
  l_person_type   VARCHAR2(1);
  l_default_schema VARCHAR2(30);
  l_groups        VARCHAR2(1000);
  l_developer_role VARCHAR2(60);
  l_description   VARCHAR2(240);
BEGIN
  APEX_UTIL.FETCH_USER(
    p_user_id                   => APEX_UTIL.GET_CURRENT_USER_ID,
    p_workspace                  => l_workspace,
    p_user_name                  => l_user_name,
    p_first_name                 => l_first_name,
    p_last_name                  => l_last_name,
    p_web_password               => l_web_password,
    p_email_address              => l_email_address,
    p_start_date                 => l_start_date,
    p_end_date                   => l_end_date,
    p_employee_id                => l_employee_id,
    p_allow_access_to_schemas    => l_allow_access_to_schemas,
    p_person_type                => l_person_type,
    p_default_schema             => l_default_schema,
    p_groups                      => l_groups,
    p_developer_role              => l_developer_role,
    p_description                => l_description);
END;
\end{verbatim}

See Also:

- "EDIT_USER Procedure"
- "GET_CURRENT_USER_ID Function"

36.30 FETCH_USER Procedure Signature 2

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.
Syntax for Signature 2

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

Parameters for Signature 2

Table 36-28  Fetch_User Parameters Signature 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_id</td>
<td>Numeric primary key of the user account</td>
</tr>
<tr>
<td>p_user_name</td>
<td>Alphanumeric name used for login.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_USERNAME Function&quot;</td>
</tr>
<tr>
<td>p_first_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_FIRST_NAME Function&quot;</td>
</tr>
<tr>
<td>p_last_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_LAST_NAME Function&quot;</td>
</tr>
<tr>
<td>p_email_address</td>
<td>Email address.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_EMAIL Function&quot;</td>
</tr>
<tr>
<td>p_groups</td>
<td>List of groups of which user is a member.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_GROUPS_USER_BELONGS_TO Function&quot; and &quot;CURRENT_USER_IN_GROUP Function&quot;</td>
</tr>
<tr>
<td>p_developer_role</td>
<td>Colon-separated list of developer roles. The following are acceptable values for this parameter:</td>
</tr>
<tr>
<td></td>
<td>null - Indicates an end user (a user who can only authenticate to developed applications).</td>
</tr>
<tr>
<td></td>
<td>Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.</td>
</tr>
<tr>
<td></td>
<td>See Also: &quot;GET_USER_ROLES Function&quot;</td>
</tr>
<tr>
<td>p_description</td>
<td>Informational</td>
</tr>
</tbody>
</table>
Example for Signature 2

The following example shows how to use the FETCH_USER procedure with Signature 2. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_user_name         VARCHAR2(100);
    l_first_name        VARCHAR2(255);
    l_last_name         VARCHAR2(255);
    l_email_address     VARCHAR2(240);
    l_groups            VARCHAR2(1000);
    l_developer_role    VARCHAR2(60);
    l_description       VARCHAR2(240);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id           => APEX_UTIL.GET_CURRENT_USER_ID,
        p_user_name         => l_user_name,
        p_first_name        => l_first_name,
        p_last_name         => l_last_name,
        p_email_address     => l_email_address,
        p_groups            => l_groups,
        p_developer_role    => l_developer_role,
        p_description       => l_description);
END;
```

See Also:

- "EDIT_USER Procedure"
- "GET_CURRENT_USER_ID Function"

36.31 FETCH_USER Procedure Signature 3

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 3

```
APEX_UTIL.FETCH_USER (  
    p_user_id                  IN                  NUMBER,  
    p_workspace                 OUT                  VARCHAR2,  
    p_user_name                 OUT                  VARCHAR2,  
    p_first_name                OUT                  VARCHAR2,  
    p_last_name                 OUT                  VARCHAR2,  
    p_web_password              OUT                  VARCHAR2,  
    p_email_address             OUT                  VARCHAR2,  
);  
```
Parameters for Signature 3

Table 36-29  Fetch_User Parameters Signature 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_id</td>
<td>Numeric primary key of the user account.</td>
</tr>
<tr>
<td>p_workspace</td>
<td>The name of the workspace.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>Alphanumeric name used for login.</td>
</tr>
<tr>
<td></td>
<td>See Also: “GET_USERNAME Function”</td>
</tr>
<tr>
<td>p_first_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: “GET_FIRST_NAME Function”</td>
</tr>
<tr>
<td>p_last_name</td>
<td>Informational.</td>
</tr>
<tr>
<td></td>
<td>See Also: “GET_LAST_NAME Function”</td>
</tr>
<tr>
<td>p_web_password</td>
<td>Obfuscated account password.</td>
</tr>
<tr>
<td>p_email_address</td>
<td>Email address.</td>
</tr>
<tr>
<td></td>
<td>See Also: “GET_EMAIL Function”</td>
</tr>
<tr>
<td>p_start_date</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_end_date</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_employee_id</td>
<td>Unused.</td>
</tr>
<tr>
<td>p_allow_access_to_schemas</td>
<td>A list of schemas assigned to the user's workspace to which user is restricted.</td>
</tr>
<tr>
<td>p_person_type</td>
<td>A database schema assigned to the user's workspace, used by default for browsing.</td>
</tr>
<tr>
<td></td>
<td>See Also: “GET_DEFAULT_SCHEMA Function”</td>
</tr>
<tr>
<td>p_default_schema</td>
<td>List of groups of which user is a member.</td>
</tr>
<tr>
<td></td>
<td>See Also: “GET_GROUPS_USER_BELONGS_TO_FUNCTION” and “CURRENT_USER_IN_GROUP_FUNCTION”</td>
</tr>
</tbody>
</table>
Table 36-29  (Cont.) Fetch_User Parameters Signature 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_developer_role</td>
<td>Colon-separated list of developer roles. The following are acceptable values for this parameter:</td>
</tr>
<tr>
<td></td>
<td>null - Indicates an end user (a user who can only authenticate to developed applications).</td>
</tr>
<tr>
<td></td>
<td>Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_prv, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.</td>
</tr>
<tr>
<td>See Also:</td>
<td>“GET_USER_ROLES Function”</td>
</tr>
<tr>
<td>p_description</td>
<td>Informational.</td>
</tr>
<tr>
<td>p_account_expiry</td>
<td>Date account password was last reset.</td>
</tr>
<tr>
<td>See Also:</td>
<td>“END_USER_ACCOUNT_DAYS_LEFT Function” and “WORKSPACE_ACCOUNT_DAYS_LEFT Function”</td>
</tr>
<tr>
<td>p_account_locked</td>
<td>Locked/Unlocked indicator Y or N.</td>
</tr>
<tr>
<td>See Also:</td>
<td>“GET_ACCOUNT_LOCKED_STATUS Function”</td>
</tr>
<tr>
<td>p_failed_access_attempts</td>
<td>Counter for consecutive login failures.</td>
</tr>
<tr>
<td>p_change_password_on_first_use</td>
<td>Setting to force password change on first use Y or N.</td>
</tr>
<tr>
<td>p_first_password_use_occurred</td>
<td>Indicates whether login with password occurred Y or N.</td>
</tr>
</tbody>
</table>

Example for Signature 3

The following example shows how to use the FETCH_USER procedure with Signature 3. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

DECLARE
l_workspace                     VARCHAR2(255);
l_user_name                     VARCHAR2(100);
l_first_name                    VARCHAR2(255);
l_last_name                     VARCHAR2(255);
l_web_password                  VARCHAR2(255);
l_email_address                 VARCHAR2(240);
l_start_date                    DATE;
l_end_date                      DATE;
l_employee_id                   NUMBER(15,0);
l_allow_access_to_schemas       VARCHAR2(4000);
l_person_type                   VARCHAR2(1);
l_default_schema                VARCHAR2(30);
l_groups                        VARCHAR2(1000);
l_developer_role                VARCHAR2(60);
l_description                   VARCHAR2(240);
l_account_expiry                DATE;
l_account_locked                VARCHAR2(1);
l_failed_access_attempts        NUMBER;
l_change_password_on_first_use  VARCHAR2(1);
l_first_password_use_occurred   VARCHAR2(1);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id                       => APEX_UTIL.GET_CURRENT_USER_ID,
        p_workspace                     => l_workspace,
        p_user_name                     => l_user_name,
        p_first_name                    => l_first_name,
        p_last_name                     => l_last_name,
        p_web_password                  => l_web_password,
        p_email_address                 => l_email_address,
        p_start_date                    => l_start_date,
        p_end_date                      => l_end_date,
        p_employee_id                   => l_employee_id,
        p_allow_access_to_schemas       => l_allow_access_to_schemas,
        p_person_type                   => l_person_type,
        p_default_schema                => l_default_schema,
        p_groups                        => l_groups,
        p_developer_role                => l_developer_role,
        p_description                   => l_description,
        p_account_expiry                => l_account_expiry,
        p_account_locked                => l_account_locked,
        p_failed_access_attempts        => l_failed_access_attempts,
        p_change_password_on_first_use  => l_change_password_on_first_use,
        p_first_password_use_occurred   => l_first_password_use_occurred);
END;

See Also:

• “EDIT_USER Procedure”
• “GET_CURRENT_USER_ID Function”

36.32 FIND_SECURITY_GROUP_ID Function

This function returns the numeric security group ID of the named workspace.
Syntax

APEX_UTIL.FIND_SECURITY_GROUP_ID(
    p_workspace    IN VARCHAR2)
RETURN NUMBER;

Parameters

Table 36-30  FIND_SECURITY_GROUP_ID Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The name of the workspace.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the FIND_SECURITY_GROUP_ID function to return the security group ID for the workspace called 'DEMOS'.

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

36.33 FIND_WORKSPACE Function

This function returns the workspace name associated with a security group ID.

Syntax

APEX_UTIL.FIND_WORKSPACE(
    p_security_group_id    IN VARCHAR2)
RETURN VARCHAR2;

Parameters

Table 36-31  FIND_WORKSPACE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_security_group_id</td>
<td>The security group ID of a workspace.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the FIND_WORKSPACE function to return the workspace name for the workspace with a security group ID of 20.

DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.FIND_WORKSPACE (p_security_group_id=>'20');
    DBMS_OUTPUT.PUT_LINE(VAL);
END;
VAL := APEX_UTIL.FIND_WORKSPACE (p_security_group_id =>'20');
END;

36.34 GET_ACCOUNT_LOCKED_STATUS Function

Returns TRUE if the account is locked and FALSE if the account is unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS(
    p_user_name IN VARCHAR2
) RETURN BOOLEAN;

Parameters

Table 36-32  GET_ACCOUNT_LOCKED_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_ACCOUNT_LOCKED_STATUS function. Use this function to check if an Application Express user account (workspace administrator, developer, or end user) in the current workspace is locked.

BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) loop
        IF APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS(p_user_name =>
            c1.user_name) THEN
            HTP.P('User Account:||c1.user_name|| is locked.');
        END IF;
        END LOOP;
END;

See Also:

- LOCK_ACCOUNT Procedure
- UNLOCK_ACCOUNT Procedure

36.35 GET_APPLICATION_STATUS Function

This function returns the current status of the application. Status values include AVAILABLE, AVAILABLE_W_EDIT_LINK, DEVELOPERS_ONLY, RESTRICTED_ACCESS, UNAVAILABLE, UNAVAILABLE_PLSQL, and UNAVAILABLE_URL.
Syntax

APEX_UTIL.GET_APPLICATION_STATUS(
    p_application_id IN NUMBER) RETURN VARCHAR2;

Parameters

Table 36-33  GET_APPLICATION_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The Application ID.</td>
</tr>
</tbody>
</table>

Example

```declare
    l_status varchar2(100);
begin
    l_status := apex_util.get_application_status(
        p_application_id => 117);
    dbms_output.put_line( 'The current application status is: ' ||
        l_status );
end;```

See Also:

"Availability" in Oracle Application Express App Builder User’s Guide

36.36 GET_ATTRIBUTE Function

This function returns the value of one of the attribute values (1 through 10) of a named user in the Application Express accounts table. Please note these are only accessible by using the APIs.

Syntax

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

Table 36-34  GET_ATTRIBUTE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>User name in the account.</td>
</tr>
<tr>
<td>p_attribute_number</td>
<td>Number of attributes in the user record (1 through 10).</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_ATTRIBUTE function to return the value for the 1st attribute for the user 'FRANK'.

DECLARE
  VAL VARCHAR2(4000);
BEGIN
  VAL := APEX_UTIL.GET_ATTRIBUTE (
    p_username => 'FRANK',
    p_attribute_number => 1);
END;

See Also:

"SET_ATTRIBUTE Procedure"

36.37 GET_AUTHENTICATION_RESULT Function

Use this function to retrieve the authentication result of the current session. Any authenticated user can call this function in a page request context.

Syntax

APEX_UTIL.GET_AUTHENTICATION_RESULT
RETURN NUMBER;

Parameters

None.

Example

The following example demonstrates how to use the post-authentication process of an application's authentication scheme to retrieve the authentication result code set during authentication.

APEX_UTIL.SET_SESSION_STATE('MY_AUTH_STATUS',
  'Authentication result:'||APEX_UTIL.GET_AUTHENTICATION_RESULT);
36.38 GET_BLOB_FILE_SRC Function

As an alternative to using the built-in methods of providing a download link, you can use the APEX_UTIL.GET_BLOB_FILE_SRC function. One advantage of this approach, is the ability to more specifically format the display of the image (with height and width tags). Please note that this approach is only valid if called from a valid Oracle Application Express session. Also, this method requires that the parameters that describe the BLOB to be listed as the format of a valid item within the application. That item is then referenced by the function.

If the URL returned by this function is passed to APEX_UTIL.PREPARE_URL, ensure to set the P.PLAIN_URL argument to TRUE to ensure that no modal dialog code is being added, when the referenced page item is on a modal page.

Syntax

APEX_UTIL.GET_BLOB_FILE_SRC (p_item_name IN VARCHAR2 DEFAULT NULL, p_v1 IN VARCHAR2 DEFAULT NULL, p_v2 IN VARCHAR2 DEFAULT NULL, p_content_disposition IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2;

Parameters

Table 36-35  GET_BLOB_FILE_SRC Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item_name</td>
<td>Name of valid application page ITEM that with type FILE that contains the source type of DB column.</td>
</tr>
<tr>
<td>p_v1</td>
<td>Value of primary key column 1.</td>
</tr>
<tr>
<td>p_v2</td>
<td>Value of primary key column 2.</td>
</tr>
<tr>
<td>p_content_disposition</td>
<td>Specify inline or attachment, all other values ignored.</td>
</tr>
</tbody>
</table>

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:

```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
FROM TEST_WITH_BLOB
```

The previous example illustrates how to display the BLOB within the report, if it can be displayed, and provide a download link, if it cannot be displayed.

---

See Also:

"About BLOB Support in Forms and Reports" in *Oracle Application Express App Builder User's Guide*

---

### 36.39 GET_BUILD_OPTION_STATUS Function Signature 1

Use this function to get the build option status of a specified application by providing the ID of the application build option.

**Syntax**

```sql
APEX_UTIL.GET_BUILD_OPTION_STATUS(
    p_application_id IN NUMBER,
    p_id              IN NUMBER;
)
```

**Parameters**

**Table 36-36  GET_BUILD_OPTION_STATUS Function Signature 1 Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application that owns the build option under shared components.</td>
</tr>
<tr>
<td>p_id</td>
<td>The ID of the build option in the application.</td>
</tr>
</tbody>
</table>
Example

The following code retrieves the current status of the specified build option that is identified by ID.

```
DECLARE
  l_status VARCHAR2(255);
BEGIN
  l_status := APEX_UTIL.GET_BUILD_OPTION_STATUS(
    P_APPLICATION_ID => 101,
    P_ID => 245935500311121039);
END;
/
```

### 36.40 GET_BUILD_OPTION_STATUS Function Signature 2

Use this function to get the build option status of a specified application by providing the name of the application build option.

**Syntax**

```
APEX_UTIL.GET_BUILD_OPTION_STATUS(
  p_application_id IN NUMBER
  p_build_option_name IN VARCHAR2);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application that owns the build option under shared components.</td>
</tr>
<tr>
<td>p_build_option_name</td>
<td>The name of the build option in the application.</td>
</tr>
</tbody>
</table>

**Example**

The following code retrieves the current status of the specified build option that is identified by name.

```
DECLARE
  l_status VARCHAR2(255);
BEGIN
  l_status := APEX_UTIL.GET_BUILD_OPTION_STATUS(
    P_APPLICATION_ID => 101,
    P_BUILD_OPTION_NAME => 'EXCLUDE_FROM_PRODUCTION');
END;
/
```
36.41 GET_CURRENT_USER_ID Function

This function returns the numeric user ID of the current user.

Syntax

APEX_UTIL.GET_CURRENT_USER_ID
RETURN NUMBER;

Parameters

None.

Example

This following example shows how to use the GET_CURRENT_USER_ID function. It returns the numeric user ID of the current user into a local variable.

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

36.42 GET_DEFAULT_SCHEMA Function

This function returns the default schema name associated with the current user.

Syntax

APEX_UTIL.GET_DEFAULT_SCHEMA
RETURN VARCHAR2;

Parameters

None.

Example

The following example shows how to use the GET_DEFAULT_SCHEMA function. It returns the default schema name associated with the current user into a local variable.

DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_DEFAULT_SCHEMA;
END;
36.43 GET_EDITION Function

This function returns the edition for the current page view.

Syntax

APEX_UTIL.GET_EDITION
RETURN VARCHAR2;

Parameters
None.

Example

The following example shows how to use the GET_EDITION function. It returns the edition name for the current page view into a local variable.

DECLARE
  VAL VARCHAR2(30);
BEGIN
  VAL := APEX_UTIL.GET_EDITION;
END;

36.44 GET_EMAIL Function

This function returns the email address associated with the named user.

Syntax

APEX_UTIL.GET_EMAIL(
  p_username IN VARCHAR2);
RETURN VARCHAR2;

Parameters

Table 36-38  GET_EMAIL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>The user name in the account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_EMAIL function to return the email address of the user 'FRANK'.

DECLARE
  VAL VARCHAR2(240);
BEGIN
36.45 GET_FEEDBACK_FOLLOW_UP Function

Use this function to retrieve any remaining follow up associated with a specific feedback.

Syntax

APEX_UTIL.GET_FEEDBACK_FOLLOW_UP (
  p_feedback_id IN NUMBER,
  p_row IN NUMBER DEFAULT 1,
  p_template IN VARCHAR2 DEFAULT '<br/>'#CREATED_ON# (#CREATED_BY#) #FOLLOW_UP#')
RETURN VARCHAR2;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_feedback_id</td>
<td>The unique identifier of the feedback item.</td>
</tr>
<tr>
<td>p_row</td>
<td>Identifies which follow-up to retrieve and is ordered by created_on_desc.</td>
</tr>
<tr>
<td>p_template</td>
<td>The template to use to return the follow up. Given the &lt;br/&gt; in the default template, the function can be used in a loop to return all the follow up to a feedback.</td>
</tr>
</tbody>
</table>

Example

The following example displays all the remaining follow-up for feedback with the ID of 123.

```
declare
  l_feedback_count number;
begin
  select count(*)
  into l_feedback_count
  from apex_team_feedback_followup
  where feedback_id = 123;

  for i in 1..l_feedback_count loop
    htp.p(apex_util.get_feedback_follow_up (  
```
36.46 GET_FILE Procedure

This procedure downloads files from the Oracle Application Express file repository. Please note if you are invoking this procedure during page processing, you must ensure that no page branch is invoked under the same condition, as it interferes with the file retrieval. This means that branches with any of the following conditions should not be set to fire:

- Branches with a 'When Button Pressed' attribute equal to the button that invokes the procedure.
- Branches with conditional logic defined that would succeed during page processing when the procedure is being invoked.
- As unconditional.

Syntax

APEX_UTIL.GET_FILE (  
  p_file_id    IN   VARCHAR2,  
  p_inline     IN   VARCHAR2 DEFAULT \"NO\");  

Parameters

Table 36-40 GET_FILE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| p_file_id   | ID in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace. The following example demonstrates how to use APEX_APPLICATION_FILES:  
  DECLARE  
  l_file_id NUMBER;  
  BEGIN  
  SELECT id  
  INTO l_file_id  
  FROM APEX_APPLICATION_FILES  
  WHERE filename = \"myxml\";  
  --  
  APEX_UTIL.GET_FILE(  
  p_file_id => l_file_id,  
  p_inline => \"YES\");  
  END; |

| p_inline    | Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment. |
Example

The following example shows how to use the GET_FILE function to return the file identified by the ID 8675309. This is displayed inline in the browser.

BEGIN
   APEX_UTIL.GET_FILE(
       p_file_id   => '8675309',
       p_inline   => 'YES');
END;

See Also:

"GET_FILE_ID Function"

36.47 GET_FILE_ID Function

This function obtains the primary key of a file in the Oracle Application Express file repository.

Syntax

APEX_UTIL.GET_FILE_ID (p_name IN VARCHAR2) RETURN NUMBER;

Parameters

Table 36-41  GET_FILE_ID Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>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.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_FILE_ID function to retrieve the database ID of the file with a filename of 'F125.sql'.

DECLARE
   l_name VARCHAR2(255);
   l_file_id NUMBER;
BEGIN
   SELECT name
       INTO l_name
   FROM APEX_APPLICATION_FILES
WHERE filename = 'F125.sql';
--
  l_file_id := APEX_UTIL.GET_FILE_ID(p_name => l_name);
END;

36.48 GET_FIRST_NAME Function

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

Syntax

APEX_UTIL.GET_FIRST_NAME
  p_username IN VARCHAR2)
RETURN VARCHAR2;

Parameters

Table 36-42  GET_FIRST_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Identifies the user name in the account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_FIRST_NAME function to return the FIRST_NAME of the user 'FRANK'.

DECLARE
  VAL VARCHAR2(255);
BEGIN
  VAL := APEX_UTIL.GET_FIRST_NAME(p_username => 'FRANK');
END;

See Also:

"SET_FIRST_NAME Procedure"

36.49 GET_GLOBAL_NOTIFICATION Function

This function gets the global notification message which is the message displayed in page #GLOBAL_NOTIFICATION# substitution string.

Syntax

APEX_UTIL.GET_GLOBAL_NOTIFICATION(
  p_application_id IN NUMBER) RETURN VARCHAR2;
Parameters

Table 36-43  GET_GLOBAL_NOTIFICATION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The Application ID.</td>
</tr>
</tbody>
</table>

Example

```sql
declare
    l_global_notification varchar2(100);
begin
    l_global_notification := apex_util.get_global_notification(
        p_application_id => 117);
    dbms_output.put_line( 'The current global notification is: ' ||
                          l_global_notification );
end;
```

See Also:

"Availability" in Oracle Application Express App Builder User's Guide

36.50 GET_GROUPS_USER_BELONGS_TO Function

This function returns a comma then a space separated list of group names to which the named user is a member.

Syntax

```sql
APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(
    p_username IN VARCHAR2
) RETURN VARCHAR2;
```

Parameters

Table 36-44  GET_GROUPS_USER_BELONGS_TO Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Identifies the user name in the account.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the `GET_GROUPS_USER_BELONGS_TO` to return the list of groups to which the user 'FRANK' is a member.

```
DECLARE
    VAL VARCHAR2(32765);
BEGIN
    VAL := APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'FRANK');
END;
```

See Also:

“EDIT_USER Procedure”

36.51 GET_GROUP_ID Function

This function returns the numeric ID of a named group in the workspace.

Syntax

```
APEX_UTIL.GET_GROUP_ID(
    p_group_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 36-45  GET_GROUP_ID Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_name</td>
<td>Identifies the user name in the account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the `GET_GROUP_ID` function to return the ID for the group named 'Managers'.

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

36.52 GET_GROUP_NAME Function

This function returns the name of a group identified by a numeric ID.
Syntax

APEX_UTIL.GET_GROUP_NAME(
    p_group_id IN NUMBER
) RETURN VARCHAR2;

Parameters

Table 36-46  GET_GROUP_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_id</td>
<td>Identifies a numeric ID of a group in the workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_GROUP_NAME function to return the name of the group with the ID 8922003.

DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;

36.53 GET_HASH Function

This function computes a hash value for all given values. Use this function to implement lost update detection for data records.

Syntax

APEX_UTIL.GET_HASH (
    p_values in apex_t_varchar2,
    p_salted in boolean default true
) RETURN VARCHAR2;

Parameters

Table 36-47  GET_HASH Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_values</td>
<td>The input values.</td>
</tr>
<tr>
<td>p_salted</td>
<td>If true (the default), salt hash with internal session information.</td>
</tr>
</tbody>
</table>

Example

declare
    l_hash varchar2(4000);
begin
    select apex_util.get_hash(apex_t_varchar2 (empno, sal, comm))
    into l_hash
    from emp
    where empno = :P1_EMPNO;

    if :P1_HASH <> l_hash then
        raise_application_error(-20001, 'Somebody already updated SAL/COMM');
    end if;

    update emp
    set sal = :P1_SAL,
         comm = :P1_COMM
    where empno = :P1_EMPNO;
exception when no_data_found then
    raise_application_error(-20001, 'Employee not found');
end;

36.54 GET_HIGH_CONTRAST_MODE_TOGGLE Function

This function returns a link to the current page that enables you to turn on or off,
toggle, the mode. For example, if you are in standard mode, this function displays a
link that when clicked switches high contrast mode on.

Syntax

APEX_UTIL.GET_HIGH_CONTRAST_MODE_TOGGLE (p_on_message  IN VARCHAR2 DEFAULT NULL,
p_off_message IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;

Parameters

Table 36-48  GET_HIGH_CONTRAST_MODE_TOGGLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_on_message</td>
<td>Optional text used for the link to switch to high contrast mode, when you are in standard mode. If this parameter is not passed, the default 'Set High Contrast Mode On' text is returned in the link.</td>
</tr>
<tr>
<td>p_off_message</td>
<td>Optional text used for the link to switch to standard mode, when you are in high contrast mode. If this parameter is not passed, the default 'Set High Contrast Mode Off' text is returned in the link.</td>
</tr>
</tbody>
</table>

Example

When running in standard mode, this function returns a link with the text 'Set High Contrast Mode On'. When the link is clicked the current page is refreshed and high contrast mode is switched on. When running in high contrast mode, a link 'Set High Contrast Mode Off' is displayed.
Contrast Mode Off is returned. When the link is clicked the current page is refreshed and switched back to standard mode.

BEGIN
    http.p(apex_util.get_high_contrast_mode_toggle);
END;

Note:

There are also 2 translatable system messages that can be overridden at application level to change the default link text that is returned for this toggle. They include:

- APEX.SET_HIGH_CONTRAST_MODE_OFF - Default text = Set High Contrast Mode Off
- APEX.SET_HIGH_CONTRAST_MODE_ON - Default text = Set High Contrast Mode On

See Also:

"SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure"

36.55 GET_LAST_NAME Function

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

Syntax

APEX_UTIL.GET_LAST_NAME(
    p_username IN VARCHAR2)
RETURN VARCHAR2;

Parameters

Table 36-49  GET_LAST_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>The user name in the user account record.</td>
</tr>
</tbody>
</table>
The following example shows how to use the function to return the \texttt{LAST\_NAME} for the user \texttt{FRANK}.

```sql
DECLARE
  VAL VARCHAR2(255);
BEGIN
  VAL := APEX_UTIL.GET\_LAST\_NAME(p\_username => 'FRANK');
END;
```

\textbf{See Also:}

“SET\_LAST\_NAME Procedure”

### 36.56 GET\_NUMERIC\_SESSION\_STATE Function

This function returns a numeric value for a numeric item. You can use this function in Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function \texttt{NV}, in place of \texttt{APEX\_UTIL.GET\_NUMERIC\_SESSION\_STATE}.

\textbf{Syntax}

```sql
APEX\_UTIL.GET\_NUMERIC\_SESSION\_STATE (p\_item IN VARCHAR2)
RETURN NUMBER;
```

\textbf{Parameters}

\textbf{Table 36-50 GET\_NUMERIC\_SESSION\_STATE Parameters}

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item</td>
<td>The case insensitive name of the item for which you want to have the session state fetched.</td>
</tr>
</tbody>
</table>

\textbf{Example}

The following example shows how to use the function to return the numeric value stored in session state for the item \texttt{my\_item}.

```sql
DECLARE
  l\_item\_value NUMBER;
BEGIN
  l\_item\_value := APEX\_UTIL.GET\_NUMERIC\_SESSION\_STATE('my\_item');
END;
```
36.57 GET_PREFERENCE Function

This function retrieves the value of a previously saved preference for a given user.

Syntax

APEX_UTIL.GET_PREFERENCE ( 
    p_preference  IN    VARCHAR2 DEFAULT NULL, 
    p_user        IN    VARCHAR2 DEFAULT USER)
RETURN VARCHAR2;

Parameters

Table 36-51    GET_PREFERENCE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_preference</td>
<td>Name of the preference to retrieve the value.</td>
</tr>
<tr>
<td>p_user</td>
<td>User for whom the preference is being retrieved.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_PREFERENCE function to return the value for the currently authenticated user's preference named default_view.

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

See Also:

• "GET_SESSION_STATE Function"
• "SET_SESSION_STATE Procedure"

See Also:

• “SET_PREFERENCE Procedure”
• "REMOVE_PREFERENCE Procedure"
• "Managing User Preferences" in Oracle Application Express Administration Guide
36.58 GET_PRINT_DOCUMENT Function Signature 1

This function returns a document as BLOB using XML based report data and RTF or XSL-FO based report layout.

Syntax

APEX_UTIL.GET_PRINT_DOCUMENT (  
  p_report_data IN BLOB,  
  p_report_layout IN CLOB,  
  p_report_layout_type IN VARCHAR2 default 'xsl-fo',  
  p_document_format IN VARCHAR2 default 'pdf',  
  p_print_server IN VARCHAR2 default NULL)  
RETURN BLOB;

Parameters

Table 36-52  GET_PRINT_DOCUMENT Signature 1 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_report_data</td>
<td>XML based report data.</td>
</tr>
<tr>
<td>p_report_layout</td>
<td>Report layout in XSL-FO or RTF format.</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4".

36.59 GET_PRINT_DOCUMENT Function Signature 2

This function returns a document as BLOB using pre-defined report query and pre-defined report layout.

Syntax

APEX_UTIL.GET_PRINT_DOCUMENT (  
  p_application_id IN NUMBER,  
  p_report_query_name IN VARCHAR2,  
  p_report_layout_name IN VARCHAR2 default null,  
  p_report_layout_type IN VARCHAR2 default 'xsl-fo',  
  p_document_format IN VARCHAR2 default 'pdf',  
  p_print_server IN VARCHAR2 default null)  
RETURN BLOB;
Parameters

Table 36-53  GET_PRINT_DOCUMENT Signature 2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>Defines the application ID of the report query.</td>
</tr>
<tr>
<td>p_report_query_name</td>
<td>Name of the report query (stored under application's shared components).</td>
</tr>
<tr>
<td>p_report_layout_name</td>
<td>Name of the report layout (stored under application's Shared Components).</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4".

36.60 GET_PRINT_DOCUMENT Function Signature 3

This function returns a document as BLOB using a pre-defined report query and RTF or XSL-FO based report layout.

Syntax

APEX_UTIL.GET_PRINT_DOCUMENT (  
  p_application_id      IN NUMBER,  
  p_report_query_name   IN VARCHAR2,  
  p_report_layout       IN CLOB,  
  p_report_layout_type  IN VARCHAR2 default 'xsl-fo',  
  p_document_format     IN VARCHAR2 default 'pdf',  
  p_print_server        IN VARCHAR2 default null)  
RETURN BLOB;

Parameters

Table 36-54  GET_PRINT_DOCUMENT Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>Defines the application ID of the report query.</td>
</tr>
<tr>
<td>p_report_query_name</td>
<td>Name of the report query (stored under application's shared components).</td>
</tr>
<tr>
<td>p_report_layout</td>
<td>Defines the report layout in XSL-FO or RTF format.</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
</tbody>
</table>
Table 36-54  (Cont.) GET_PRINT_DOCUMENT Signature 3 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4".

36.61 GET_PRINT_DOCUMENT Function Signature 4

This function returns a document as BLOB using XML based report data and RTF or XSL-FO based report layout.

Syntax

APEX_UTIL.GET_PRINT_DOCUMENT (  
  p_report_data       IN CLOB,  
  p_report_layout     IN CLOB,  
  p_report_layout_type IN VARCHAR2 default 'xsl-fo',  
  p_document_format   IN VARCHAR2 default 'pdf',  
  p_print_server      IN VARCHAR2 default NULL)  
RETURN BLOB;

Parameters

Table 36-55  GET_PRINT_DOCUMENT Signature 4 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_report_data</td>
<td>XML based report data, must be encoded in UTF-8.</td>
</tr>
<tr>
<td>p_report_layout</td>
<td>Report layout in XSL-FO or RTF format.</td>
</tr>
<tr>
<td>p_report_layout_type</td>
<td>Defines the report layout type, that is &quot;xsl-fo&quot; or &quot;rtf&quot;.</td>
</tr>
<tr>
<td>p_document_format</td>
<td>Defines the document format, that is &quot;pdf&quot;, &quot;rtf&quot;, &quot;xls&quot;, &quot;htm&quot;, or &quot;xml&quot;.</td>
</tr>
<tr>
<td>p_print_server</td>
<td>URL of the print server. If not specified, the print server is derived from preferences.</td>
</tr>
</tbody>
</table>

Example for Signature 4

The following example shows how to use the GET_PRINT_DOCUMENT using Signature 4 (Document returns as a BLOB using XML based report data and RTF or XSL-FO based report layout). In this example, GET_PRINT_DOCUMENT is used with APEX_MAIL.SEND and APEX_MAIL.ADD_ATTACHMENT to send an email with an attachment of the file returned by GET_PRINT_DOCUMENT. Both the report data and layout are taken from values stored in page items (P1_XML and P1_XSL).

DECLARE
  l_id number;
  l_document BLOB;
36.62 GET_SCREEN_READER_MODE_TOGGLE Function

This function returns a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches screen reader mode on.

Syntax

APEX_UTIL.GET_SCREEN_READER_MODE_TOGGLE (    p_on_message  IN VARCHAR2 DEFAULT NULL,    p_off_message IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2;

Parameters

Table 36-56  GET_SCREEN_READER_MODE_TOGGLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_on_message</td>
<td>Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text is returned in the link.</td>
</tr>
<tr>
<td>p_off_message</td>
<td>Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text is returned in the link.</td>
</tr>
</tbody>
</table>

Example

When running in standard mode, this function returns a link with the text 'Set Screen Reader Mode On'. When the link is clicked the current page is refreshed and screen
reader mode is switched on. When running in screen reader mode, a link ‘Set Screen Reader Mode Off’ is returned. When the link is clicked the current page is refreshed and switched back to standard mode.

BEGIN
    htp.p(apex_util.get_screen_reader_mode_toggle);
END;

### See Also:
"SHOW_SCREEN_READER_MODE_TOGGLE Procedure"

## 36.63 GET_SESSION_LANG Function

This function returns the language setting for the current user in the current Application Express session.

### Syntax

APEX_UTIL.GET_SESSION_LANG
RETURN VARCHAR2;

### Parameters

None.

### Example

The following example shows how to use the GET_SESSION_LANG function. It returns the session language for the current user in the current Application Express session into a local variable.

DECLARE
    VAL VARCHAR2(5);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_LANG;
END;

## 36.64 GET_SESSION_STATE Function

This function returns the value for an item. You can use this function in your Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function V, in place of APEX_UTIL.GET_SESSION_STATE.
Syntax

APEX_UTIL.GET_SESSION_STATE (  
   p_item  IN   VARCHAR2)  
RETURN VARCHAR2;

Parameters

Table 36-57   GET_SESSION_STATE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_item</td>
<td>The case insensitive name of the item for which you want to have the session state fetched.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_SESSION_STATE function to return the value stored in session state for the item 'my_item'.

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

See Also:

- “GET_NUMERIC_SESSION_STATE Function”
- “SET_SESSION_STATE Procedure”

36.65 GET_SESSION_TERRITORY Function

This function returns the territory setting for the current user in the current Application Express session.

Syntax

APEX_UTIL.GET_SESSION_TERRITORY 
RETURN VARCHAR2;

Parameters

None.
Example

The following example shows how to use the GET_SESSION_TERRITORY function. It returns the session territory setting for the current user in the current Application Express session into a local variable.

DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_TERRITORY;
END;

36.66 GET_SESSION_TIME_ZONE Function

This function returns the time zone for the current user in the current Application Express session. This value is null if the time zone is not explicitly set by using APEX_UTIL.SET_SESSION_TIME_ZONE or if an application's automatic time zone attribute is enabled.

Syntax

APEX_UTIL.GET_SESSION_TIME_ZONE
RETURN VARCHAR2;

Parameters

None.

Example

The following example shows how to use the GET_SESSION_TIME_ZONE function. It returns the session time zone for the current user in the current Application Express session into a local variable.

BEGIN
    VAL := APEX_UTIL.GET_SESSION_TIME_ZONE;
END;

36.67 GET_SINCE Function

This function returns the relative date in words (for example, 2 days from now, 30 minutes ago). It also accepts a second optional p_short parameter and returns "in 2d", "30m". This function is equivalent to using the SINCE and SINCE_SHORT format mask available within Oracle Application Express and is useful within SQL queries or PL/SQL routines.

Syntax

APEX_UTIL.GET_SINCE ( p_date date )
p_short in [ boolean default false | varchar2 default 'N' ]
RETURN VARCHAR2;

Parameters

Table 36-58  GET_SINCE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_date</td>
<td>The date you want formatted.</td>
</tr>
<tr>
<td>p_short</td>
<td>Boolean or 'Y' / 'N' to indicate whether to return a short version of relative date.</td>
</tr>
</tbody>
</table>

Example

select application_id, application_name, apex_util.get_since(last_updated_on) last_update
from apex_applications
order by application_id

Syntax

APEX_UTIL.GET_SINCE ( p_value in [ timestamp | timestamp with time zone | timestamp with local time zone ],
           p_short in [ boolean default false | varchar2 default 'N' ] )
RETURN VARCHAR2;

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>The TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE you want formatted.</td>
</tr>
<tr>
<td>p_short</td>
<td>Boolean or 'Y' / 'N' to indicate whether to return a short version of relative date.</td>
</tr>
</tbody>
</table>

Example

This returns the LAST_UPDATE column with the normal formatting.

select application_id, application_name,
apex_util.get_since( last_updated_on ) last_update
from apex_applications
order by application_id;

This returns the LAST_UPDATE column with the short formatting.

select application_id, application_name,
apex_util.get_since( last_updated_on, p_short => 'Y' ) last_update
from apex_applications
order by application_id

36.68 GET_SUPPORTING_OBJECT_SCRIPT Function

This function gets supporting object scripts defined in an application.

Note:
The workspace ID must be set before the call.

Syntax

APEX_UTIL.GET_SUPPORTING_OBJECT_SCRIPT ( 
    p_application_id  in number, 
    p_script_type     in varchar2 ) return clob;

Parameters

Table 36-59  GET_SUPPORTING_OBJECT_SCRIPT Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID to get supporting objects from.</td>
</tr>
<tr>
<td>p_script_type</td>
<td>The supporting objects script type. Valid values are apex_util.c_install_script, apex_util.c_upgrade_script, apex_util.c_deinstall_script.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to set workspace ID for workspace FRED, then get supporting objects from application ID 100.

```
declare
    l_install_script   clob;
    l_upgrade_script   clob;
    l_deinstall_script clob;
begin
    apex_util.set_workspace( p_workspace => 'FRED' );

    l_install_script :=
    apex_util.get_supporting_object_script( p_application_id => 100,
                                           p_script_type => apex_util.c_install_script );

    l_upgrade_script :=
    apex_util.get_supporting_object_script( p_application_id => 100,
                                            p_script_type => apex_util.c_upgrade_script );

    l_deinstall_script :=
    apex_util.get_supporting_object_script( p_application_id => 100,
                                             p_script_type => apex_util.c_deinstall_script );
```
36.69 GET_SUPPORTING_OBJECT_SCRIPT Procedure

This procedure gets supporting object scripts and outputs to `sys.dbms_output` buffer or download as a file.

Note:
The workspace ID must be set before the call.

Syntax

```
APEX_UTIL.GET_SUPPORTING_OBJECT_SCRIPT(
    p_application_id  in number,
    p_script_type     in varchar2,
    p_output_type     in varchar2 default c_output_as_dbms_output );
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The application ID to get supporting objects from.</td>
</tr>
<tr>
<td>p_script_type</td>
<td>The supporting objects script type. Valid values are <code>apex_util.c_install_script</code>, <code>apex_util.c_upgrade_script</code>, <code>apex_util.c_deinstall_script</code>.</td>
</tr>
<tr>
<td>p_output_type</td>
<td>The script can be output to <code>sys.dbms_output</code> buffer or download as a file. Valid values are <code>apex_util.c_output_as_dbms_output</code>, <code>apex_util.c_output_as_file</code>. The default is <code>c_output_as_dbms_output</code>.</td>
</tr>
</tbody>
</table>

Examples

The following example shows how to set workspace ID for workspace FRED, then get install script from application ID 100 and output to the command-line buffer.

```
set serveroutput on;
begin
    apex_util.set_workspace( p_workspace => 'FRED' );
    apex_util.get_supporting_object_script( 
        p_application_id => 100,
        p_script_type => apex_util.c_install_script );
end;
```
The following example shows how to download upgrade script file from application ID 100 in the browser. Useful if the script needs to be downloaded using an application process.

begin
    apex_util.set_workspace( p_workspace => 'FRED');
    apex_util.get_supporting_object_script( 
        p_application_id => 100,
        p_script_type    => apex_util.c_upgrade_script,
        p_output_type    => apex_util.c_output_as_file );
end;

36.70 GET_USER_ID Function

This function returns the numeric ID of a named user in the workspace.

Syntax

APEX_UTIL.GET_USER_ID(
    p_username   IN VARCHAR2)
RETURN NUMBER;

Parameters

Table 36-61   GET_USER_ID Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Identifies the name of a user in the workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_USER_ID function to return the ID for the user named 'FRANK'.

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

36.71 GET_USER_ROLES Function

This function returns the DEVELOPER_ROLE field stored in the named user account record. Please note that currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.
Syntax

APEX_UTIL.GET_USER_ROLES(
    p_username IN VARCHAR2)
RETURN VARCHAR2;

Parameters

Table 36-62  GET_USER_ROLES Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Identifies a user name in the account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_USER_ROLES function to return colon separated list of roles stored in the DEVELOPER_ROLE field for the user 'FRANK'.

DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.GET_USER_ROLES(p_username=>'FRANK');
END;

36.72 GET_USERNAME Function

This function returns the user name of a user account identified by a numeric ID.

Syntax

APEX_UTIL.GET_USERNAME(
    p_userid IN NUMBER)
RETURN VARCHAR2;

Parameters

Table 36-63  GET_USERNAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_userid</td>
<td>Identifies the numeric ID of a user account in the workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the GET_USERNAME function to return the user name for the user with an ID of 228922003.

DECLARE
    VAL VARCHAR2(100);
BEGIN

36.73 HOST_URL Function

This function returns the URL to the Application Express instance, depending on the option passed.

Syntax

APEX_UTIL.HOST_URL (  
   p_option IN VARCHAR2 DEFAULT NULL)  
RETURN VARCHAR2;

Parameters

Table 36-64   HOST_URL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_option</td>
<td>Specifies the parts of the URL to include.</td>
</tr>
<tr>
<td></td>
<td>Possible values for p_option include:</td>
</tr>
<tr>
<td></td>
<td>• NULL - Return URL up to port number. For example:</td>
</tr>
<tr>
<td></td>
<td><a href="http://myserver.com:7778">http://myserver.com:7778</a></td>
</tr>
<tr>
<td></td>
<td>• SCRIPT - Return URL to include script name. For example:</td>
</tr>
<tr>
<td></td>
<td><a href="https://myserver.com:7778/pls/apex/">https://myserver.com:7778/pls/apex/</a></td>
</tr>
<tr>
<td></td>
<td>• IMGPRE - Return URL to include image prefix. For example:</td>
</tr>
<tr>
<td></td>
<td><a href="https://myserver.com:7778/i/">https://myserver.com:7778/i/</a></td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the HOST_URL function to return the URL, including the script name, to the current Application Express instance.

declare
   l_host_url      varchar2(4000);
   l_url           varchar2(4000);
   l_application   varchar2(30) := 'f?p=100:1';
   l_email_body    varchar2(32000);
begin
   l_host_url := apex_util.host_url('SCRIPT');
   l_url := l_host_url||l_application;
36.74 HTML_PCT_GRAPH_MASK Function

Use this function to scale a graph. This function can also be used by classic and interactive reports with format mask of GRAPH. This generates a <div> tag with inline styles.

Syntax

APEX_UTIL.HTML_PCT_GRAPH_MASK (  
    p_number IN NUMBER DEFAULT NULL,  
    p_size IN NUMBER DEFAULT 100,  
    p_background IN VARCHAR2 DEFAULT NULL,  
    p_bar_background IN VARCHAR2 DEFAULT NULL,  
    p_format IN VARCHAR2 DEFAULT NULL)  
RETURN VARCHAR2;

Parameters

Table 36-65  HTML_PCT_GRAPH_MASK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_number</td>
<td>Number between 0 and 100.</td>
</tr>
<tr>
<td>p_size</td>
<td>Width of graph in pixels.</td>
</tr>
<tr>
<td>p_background</td>
<td>Six character hexadecimal background color of chart bar (not bar color).</td>
</tr>
<tr>
<td>p_bar_background</td>
<td>Six character hexadecimal background color of chart bar (bar color).</td>
</tr>
<tr>
<td>p_format</td>
<td>If this parameter is supplied, p_size, p_background and p_bar_background are ignored.</td>
</tr>
<tr>
<td></td>
<td>This parameter uses the following format:</td>
</tr>
<tr>
<td></td>
<td>PCT_GRAPH:&lt;BACKGROUND&gt;:&lt;FOREGROUND&gt;:&lt;CHART_WIDTH&gt;</td>
</tr>
<tr>
<td></td>
<td>position 1: PCT_GRAPH format mask indicator</td>
</tr>
<tr>
<td></td>
<td>position 2: Background color in hexadecimal, 6 characters (optional)</td>
</tr>
<tr>
<td></td>
<td>position 3: Foreground &quot;bar&quot; color in hexadecimal, 6 characters (optional)</td>
</tr>
<tr>
<td></td>
<td>position 4: Chart width in pixels. Numeric and defaults to 100.</td>
</tr>
<tr>
<td></td>
<td>p_number is automatically scaled so that 50 is half of chart_width (optional).</td>
</tr>
</tbody>
</table>

Example

The following is an SQL example.

select apex_util.html_pct_graph_mask(33) from dual

The following is a report numeric column format mask example.

PCT_GRAPH:777777:111111:200
36.75 INCREMENT_CALENDAR Procedure

Use this procedure to navigate to the next set of days in the calendar. Depending on what the calendar view is, this procedure navigates to the next month, week or day. If it is a Custom Calendar the total number of days between the start date and end date are navigated.

Syntax

APEX_UTIL.INCREMENT_CALENDAR;

Parameter

None.

Example

In this example, if you create a button called NEXT in the Calendar page and create a process that fires when the create button is clicked the following code navigates the calendar.

APEX_UTIL.INCREMENT_CALENDAR

36.76 IR_CLEAR Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure clears report settings.

Note:

This procedure should be used only in a page submit process.

Syntax

APEX_UTIL.IR_CLEAR(
    p_page_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
### Parameters

#### Table 36-66  IR_CLEAR Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_report_alias</td>
<td>Identifies the saved report alias within the current application page. To clear a Primary report, <code>p_report_alias</code> must be ‘PRIMARY’ or leave as NULL. To clear a saved report, <code>p_report_alias</code> must be the name of the saved report. For example, to clear report ‘1234’, <code>p_report_alias</code> must be ‘1234’.</td>
</tr>
</tbody>
</table>

#### Example

The following example shows how to use the IR_CLEAR procedure to clear Interactive report settings with alias of ‘8101021' in page 1 of the current application.

```sql
BEGIN
   APEX_UTIL.IR_CLEAR(
      p_page_id => 1,
      p_report_alias => '8101021'
   )
END;
```

### 36.77 IR_DELETE_REPORT Procedure [DEPRECATED]

---

**Note:**

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure deletes saved Interactive reports. It deletes all saved reports except the Primary Default report.

#### Syntax

```sql
APEX_UTIL.IR_DELETE_REPORT(
   p_report_id IN NUMBER);
```

#### Parameters

#### Table 36-67  IR_DELETE_REPORT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_report_id</td>
<td>Report ID to delete within the current Application Express application.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the `IR_DELETE_REPORT` procedure to delete the saved Interactive report with ID of '880629800374638220' in the current application.

```sql
BEGIN
    APEX_UTIL.IR_DELETE_REPORT(
        p_report_id => '880629800374638220');
END;
```

36.78 IR_DELETE_SUBSCRIPTION Procedure [DEPRECATED]

Note:

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure deletes Interactive subscriptions.

Syntax

```sql
APEX_UTIL.IR_DELETE_SUBSCRIPTION(
    p_subscription_id IN NUMBER);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_subscription_id</td>
<td>Subscription ID to delete within the current workspace.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the `IR_DELETE_SUBSCRIPTION` procedure to delete the subscription with ID of '880629800374638220' in the current workspace.

```sql
BEGIN
    APEX_UTIL.IR_DELETE_SUBSCRIPTION(
        p_subscription_id => '880629800374638220');
END;
```
36.79 IR_FILTER Procedure [DEPRECATED]

Note:
The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure creates a filter on an interactive report.

Note:
This procedure should be used only in a page submit process.

Syntax

APEX_UTIL.IR_FILTER(
    p_page_id       IN NUMBER,
    p_report_column IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_filter_value  IN VARCHAR2,
    p_report_alias  IN VARCHAR2 DEFAULT NULL);

Parameters

Table 36-69  IR_FILTER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_report_column</td>
<td>Name of the report SQL column, or column alias, to be filtered.</td>
</tr>
<tr>
<td>p_operator_abbr</td>
<td>Filter type. Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td>• EQ = Equals</td>
</tr>
<tr>
<td></td>
<td>• NEQ = Not Equals</td>
</tr>
<tr>
<td></td>
<td>• LT = Less than</td>
</tr>
<tr>
<td></td>
<td>• LTE = Less than or equal to</td>
</tr>
<tr>
<td></td>
<td>• GT = Greater Than</td>
</tr>
<tr>
<td></td>
<td>• GTE = Greater than or equal to</td>
</tr>
<tr>
<td></td>
<td>• LIKE = SQL Like operator</td>
</tr>
<tr>
<td></td>
<td>• N = Null</td>
</tr>
<tr>
<td></td>
<td>• NN = Not Null</td>
</tr>
<tr>
<td></td>
<td>• C = Contains</td>
</tr>
<tr>
<td></td>
<td>• NC = Not Contains</td>
</tr>
<tr>
<td></td>
<td>• IN = SQL In Operator</td>
</tr>
<tr>
<td></td>
<td>• NIN = SQL Not In Operator</td>
</tr>
</tbody>
</table>
Table 36-69  (Cont.) IR_FILTER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_filter_value</td>
<td>Filter value. This value is not used for 'N' and 'NN'.</td>
</tr>
<tr>
<td>p_report_alias</td>
<td>Identifies the saved report alias within the current application page. To create a filter on a Primary report, p_report_alias must be 'PRIMARY' or leave as NULL. To create a filter on a saved report, p_report_alias must be the name of the saved report. For example, to create a filter on report '1234', p_report_alias must be '1234'.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the IR_FILTER procedure to filter Interactive report with alias of '8101021' in page 1 of the current application with DEPTNO equals 30.

BEGIN
APEX_UTIL.IR_FILTER(
    p_page_id        => 1,
    p_report_column  => 'DEPTNO',
    p_operator_abbr  => 'EQ',
    p_filter_value   => '30',
    p_report_alias   => '8101021'
);
END;

36.80 IR_RESET Procedure [DEPRECATED]

Note:
The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure resets report settings back to the default report settings. Resetting a report removes any customizations you have made.

Note:
This procedure should be used only in a page submit process.
Syntax

APEX_UTIL.IR_RESET(
    p_page_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);

Parameters

Table 36-70  IR_RESET Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_page_id</td>
<td>Page of the current Application Express application that contains an interactive report.</td>
</tr>
<tr>
<td>p_report_alias</td>
<td>Identifies the saved report alias within the current application page. To reset a Primary report, p_report_alias must be 'PRIMARY' or leave as NULL. To reset a saved report, p_report_alias must be the name of the saved report. For example, to reset report '1234', p_report_alias must be '1234'.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the IR_RESET procedure to reset Interactive report settings with alias of '8101021' in page 1 of the current application.

```sql
BEGIN
    APEX_UTIL.IR_RESET(
        p_page_id => 1,
        p_report_alias => '8101021' );
END;
```

36.81 IS_HIGH_CONTRAST_SESSION Function

This function returns a boolean TRUE if the session is in high contrast mode and returns a boolean FALSE if not in high contrast mode.

Syntax

APEX_UTIL.IS_HIGH_CONTRAST_SESSION
RETURN BOOLEAN;

Parameters

None.
Example

In this example, if the current session is running in high contrast mode, a high contrast specific CSS file 'my_app_hc.css' is added to the HTML output of the page.

BEGIN
    IF apex_util.is_high_contrast_session THEN
        apex_css.add_file (
            p_name => 'my_app_hc');
    END IF;
END;

36.82 IS_HIGH_CONTRAST_SESSION_YN Function

This function returns Y if the session is in high contrast mode and N if not in high contrast mode.

Syntax

APEX_UTIL.IS_HIGH_CONTRAST_SESSION_YN
RETURN VARCHAR2;

Parameters

None.

Example

In this example, if the current session is running in high contrast mode, a high contrast specific CSS file, my_app_hc.css, is added to the HTML output of the page.

BEGIN
    IF apex_util.is_high_contrast_session_yn = 'Y' THEN
        apex_css.add_file (
            p_name => 'my_app_hc');
    END IF;
END;

36.83 IS_LOGIN_PASSWORD_VALID Function

This function returns a Boolean result based on the validity of the password for a named user account in the current workspace. This function returns TRUE if the password matches and it returns FALSE if the password does not match.

Syntax

APEX_UTIL.IS_LOGIN_PASSWORD_VALID(
    p_username IN VARCHAR2 default null,
    p_password IN VARCHAR2 default null)
RETURN BOOLEAN;
Parameters

Table 36-71  IS_LOGIN_PASSWORD_VALID Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>User name in account.</td>
</tr>
<tr>
<td>p_password</td>
<td>Password to be compared with password stored in the account.</td>
</tr>
</tbody>
</table>

Returns

- **true**: The user credentials are valid.
- **false**: The user credentials are invalid.
- **null**: Credentials checking was delayed because of too many wrong combinations.

Example

The following example shows how to use the IS_LOGIN_PASSWORD_VALID function to check if the user 'FRANK' has the password 'tiger'. TRUE is returned if this is a valid password for 'FRANK', FALSE is returned if not.

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

36.84 IS_SCREEN_READER_SESSION Function

This function returns a boolean TRUE if the session is in screen reader mode and returns a boolean FALSE if not in screen reader mode.

Syntax

```sql
APEX_UTIL.IS_SCREEN_READER_SESSION RETURN BOOLEAN;
```

Parameters

None

Example

```sql
BEGIN
    IF apex_util.is_screen_reader_session then
        htp.p('Screen Reader Mode');
    END;
```
36.85 IS_SCREEN_READER_SESSION_YN Function

This function returns ‘Y’ if the session is in screen reader mode and ‘N’ if not in screen reader mode.

Syntax

APEX_UTIL.IS_SCREEN_READER_SESSION_YN
RETURN VARCHAR2;

Parameters

None

Example

BEGIN
    IF apex_util.is_screen_reader_session_yn = 'Y' then
        htp.p('Screen Reader Mode');
    END IF;
END;

36.86 IS_USERNAME_UNIQUE Function

This function returns a Boolean result based on whether the named user account is unique in the workspace.

Syntax

APEX_UTIL.IS_USERNAME_UNIQUE(
    p_username IN VARCHAR2)
RETURN BOOLEAN;

Parameters

Table 36-72  IS_USERNAME_UNIQUE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Identifies the user name to be tested.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the IS_USERNAME_UNIQUE function. If the user 'FRANK' already exists in the current workspace, FALSE is returned, otherwise TRUE is returned.

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

36.87 KEYVAL_NUM Function

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

Syntax

APEX_UTIL.KEYVAL_NUM
RETURN NUMBER;

Parameters

None

Example

The following example shows how to use the KEYVAL_NUM function to return the current value of the package variable apex_utilities.g_val_num.

DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.KEYVAL_NUM;
END;

See Also:

"SAVEKEY_NUM Function"

36.88 KEYVAL_VC2 Function

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

APEX_UTIL.KEYVAL_VC2;

Parameters

None.

Example

The following example shows how to use the KEYVAL_VC2 function to return the current value of the package variable apex_utilities.g_val_vc2.

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

See Also:

"SAVEKEY_VC2 Function"

36.89 LOCK_ACCOUNT Procedure

Sets a user account status to locked. Must be run by an authenticated workspace administrator in the context of a page request.

Syntax

APEX_UTIL.LOCK_ACCOUNT (p_user_name IN VARCHAR2);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the LOCK_ACCOUNT procedure. Use this procedure to lock an Application Express account (workspace administrator,
developer, or end user) in the current workspace. This action locks the account for use by administrators, developers, and end users.

BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.LOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:  |c1.user_name| is now locked.');
    END LOOP;
END;

### See Also:

- "UNLOCK_ACCOUNT Procedure"
- "GET_ACCOUNT_LOCKED_STATUS Function"

---

### 36.90 PASSWORD_FIRST_USE_OCCURRED Function

Returns TRUE if the account's password has changed since the account was created, an Oracle Application Express administrator performs a password reset operation that results in a new password being emailed to the account holder, or a user has initiated password reset operation. This function returns FALSE if the account's password has not been changed since either of the events just described.

This function may be run in a page request context by any authenticated user.

**Syntax**

APEX_UTIL.PASSWORD_FIRST_USE_OCCURRED ( 
    p_user_name IN VARCHAR2) 
RETURN BOOLEAN;

**Parameters**

<table>
<thead>
<tr>
<th>Table 36-74 PASSWORD_FIRST_USE_OCCURRED Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>p_user_name</td>
</tr>
</tbody>
</table>

**Example**

The following example shows how to use the PASSWORD_FIRST_USE_OCCURRED function. Use this function to check if the password for an Application Express user account (workspace administrator, developer, or end user) in the current workspace has been changed by the user the first time the user logged in after the password was initially set during account creation, or was changed by one of the password reset operations.
described above. This is meaningful only with accounts for which the
CHANGE_PASSWORD_ON_FIRST_USE attribute is set to Yes.

BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        IF APEX_UTIL.PASSWORD_FIRST_USE_OCCURRED(p_user_name => c1.user_name) THEN
            htp.p('User: ' || c1.user_name || ' has logged in and updated the
password.');
        END IF;
    END LOOP;
END;

See Also:
"CHANGE_PASSWORD_ON_FIRST_USE Function"

36.91 PREPARE_URL Function

The PREPARE_URL function serves two purposes:

1. To return an f?p URL with the Session State Protection checksum argument
   (&cs=) if one is required.

2. To return an f?p URL with the session ID component replaced with zero (0) if the
   zero session ID feature is in use and other criteria are met.

Note:
The PREPARE_URL functions returns the f?p URL with &cs=<large hex value>
appended. If you use this returned value, for example in JavaScript, it may
be necessary to escape the ampersand in the URL to conform with syntax
rules of the particular context.

Syntax

APEX_UTIL.PREPARE_URL ( 
    p_url                    IN VARCHAR2,
    p_url_charset            IN VARCHAR2 default null,
    p_checksum_type          IN VARCHAR2 default null,
    p_triggering_element     IN VARCHAR2 default 'this'
    p_plain_url              IN BOOLEAN  default false
) RETURN VARCHAR2;
Parameters

Table 36-75  PREPARE_URL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>An f?p relative URL with all substitutions resolved.</td>
</tr>
<tr>
<td>p_url_charset</td>
<td>The character set name (for example, UTF-8) to use when escaping special characters contained within argument values.</td>
</tr>
<tr>
<td>p_checksum_type</td>
<td>Null or any of the following six values, SESSION or 3, PRIVATE_BOOKMARK or 2, or PUBLIC_BOOKMARK or 1.</td>
</tr>
<tr>
<td>p_triggering_element</td>
<td>A jQuery selector (for example, #my_button, where my_button is the static ID for a button element), to identify which element to use to trigger the dialog. This is required for Modal Dialog support.</td>
</tr>
<tr>
<td>p_plain_url</td>
<td>Exclude any of the generated JavaScript code to close a modal dialog, if applicable. By default, if this function is called from a modal dialog, JavaScript code to close the modal dialog is included in the generated URL.</td>
</tr>
</tbody>
</table>

Example 1

The following example shows how to use the PREPARE_URL function to return a URL with a valid 'SESSION' level checksum argument. This URL sets the value of P1_ITEM page item to xyz.

DECLARE
    l_url varchar2(2000);
    l_app number := v('APP_ID');
    l_session number := v('APP_SESSION');
BEGIN
    l_url := APEX_UTIL.PREPARE_URL(
        p_url => 'f?p= ' || l_app || ':1:' || l_session || '::NO::P1_ITEM:xyz',
        p_checksum_type => 'SESSION');
END;

Example 2

The following example shows how to use the PREPARE_URL function to return a URL with a zero session ID. In a PL/SQL Dynamic Content region that generates f?p URLs (anchors), call PREPARE_URL to ensure that the session ID is set to zero when the zero session ID feature is in use, when the user is a public user (not authenticated), and when the target page is a public page in the current application:

http.p(APEX_UTIL.PREPARE_URL(p_url => 'f?p= ' || :APP_ID || ':10:' || :APP_SESSION || '::NO::P10_ITEM:ABC'));

When using PREPARE_URL for this purpose, the p_url_charset and p_checksum_type arguments can be omitted. However, it is permissible to use them when both the Session State Protection and Zero Session ID features are applicable.
36.92 PUBLIC_CHECK_AUTHORIZATION Function [DEPRECATED]

Given the name of a authorization scheme, this function determines if the current user passes the security check.

Syntax

APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION (  
    p_security_scheme IN VARCHAR2)  
RETURN BOOLEAN;

Parameters

Table 36-76 PUBLIC_CHECK_AUTHORIZATION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_security_name</td>
<td>The name of the authorization scheme that determines if the user passes the security check.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the PUBLIC_CHECK_AUTHORIZATION function to check if the current user passes the check defined in the my_auth_scheme authorization scheme.

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

36.93 PURGE_REGIONS_BY_APP Procedure

Deletes all cached regions for an application.
Syntax

APEX_UTIL.PURGE_REGIONS_BY_APP (  
    p_application IN NUMBER);

Parameters

Table 36-77  PURGE_REGIONS_BY_APP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
</tbody>
</table>

Example

The following example show how to use APEX_UTIL.PURGE_REGIONS_BY_APP to delete all cached regions for application #123.

BEGIN  
    APEX_UTIL.PURGE_REGIONS_BY_APP(p_application=>123);  
END;

36.94 PURGE_REGIONS_BY_NAME Procedure

Deletes all cached values for a region identified by the application ID, page number and region name.

Syntax

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

Parameters

Table 36-78  PURGE_REGIONS_BY_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
<tr>
<td>p_page</td>
<td>The number of the page containing the region to be deleted.</td>
</tr>
<tr>
<td>p_region_name</td>
<td>The region name to be deleted.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the `PURGE_REGIONS_BY_NAME` procedure to delete all the cached values for the region 'my_cached_region' on page 1 of the current application.

```sql
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_NAME(
        p_application => :APP_ID,
        p_page => 1,
        p_region_name => 'my_cached_region');
END;
```

36.95 PURGE_REGIONS_BY_PAGE Procedure

Deletes all cached regions by application and page.

Syntax

```sql
APEX_UTIL.PURGE_REGIONS_BY_PAGE (
    p_application IN NUMBER,
    p_page        IN NUMBER);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application</td>
<td>The identification number (ID) of the application.</td>
</tr>
<tr>
<td>p_page</td>
<td>The identification number of page containing the region.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the `PURGE_REGIONS_BY_PAGE` procedure to delete all the cached values for regions on page 1 of the current application.

```sql
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_PAGE(
        p_application => :APP_ID,
        p_page => 1);
END;
```

36.96 REDIRECT_URL Procedure

This procedure calls `owa_util.redirect_url` to tell the browser to redirect to a new URL. Afterwards, it automatically calls `apex_application.stop_apex_engine` to abort further processing of the Application Express application.
Syntax

APEX_UTIL.REDIRECT_URL (  
    p_url in varchar2,  
    p_reset_htp_buffer in boolean default true );

Parameters

Table 36-80  REDIRECT_URL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The URL the browser requests.</td>
</tr>
<tr>
<td>p_reset_htp_buffer</td>
<td>Set to TRUE to reset the HTP buffer to make sure the browser understands the redirect to the new URL and is not confused by data that is already written to the HTP buffer. Set to FALSE if the application has it’s own cookie to use in the response.</td>
</tr>
</tbody>
</table>

Example

The following example tells the browser to redirect to http://www.oracle.com and immediately stops further processing.

apex_util.redirect_url (  
    p_url => 'http://www.oracle.com/');

36.97 REMOVE_PREFERENCE Procedure

This procedure removes the preference for the supplied user.

Syntax

APEX_UTIL.REMOVE_PREFERENCE (  
    p_preference IN VARCHAR2 DEFAULT NULL,  
    p_user IN VARCHAR2 DEFAULT V('USER'));

Parameters

Table 36-81  REMOVE_PREFERENCE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_preference</td>
<td>Name of the preference to remove.</td>
</tr>
<tr>
<td>p_user</td>
<td>User for whom the preference is defined.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the `REMOVE_PREFERENCE` procedure to remove the preference `default_view` for the currently authenticated user.

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

See Also:

- "GET_PREFERENCE Function"
- "SET_PREFERENCE Procedure"
- "Managing User Preferences" in Oracle Application Express Administration Guide

36.98 REMOVE_SORT_PREFERENCES Procedure

This procedure removes the user's column heading sorting preference value.

Syntax

APEX_UTIL.REMOVE_SORT_PREFERENCES (  
    p_user  IN   VARCHAR2 DEFAULT V('USER'));

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user</td>
<td>Identifies the user for whom sorting preferences are removed.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the `REMOVE_SORT_PREFERENCES` procedure to remove the currently authenticated user's column heading sorting preferences.

BEGIN
    APEX_UTIL.REMOVE_SORT_PREFERENCES(:APP_USER);
END;
36.99 REMOVE_USER Procedure

This procedure removes the user account identified by the primary key or a user name. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

APEX_UTIL.REMOVE_USER(
    p_user_id   IN NUMBER,
    p_user_name IN VARCHAR2);

Parameters

Table 36-83  REMOVE_USER Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_id</td>
<td>The numeric primary key of the user account record.</td>
</tr>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following examples show how to use the REMOVE_USER procedure to remove a user account. Firstly, by the primary key (using the p_user_id parameter) and secondly by user name (using the p_user_name parameter).

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

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

36.100 RESET_AUTHORIZATIONS Procedure [DEPRECATED]

Note:

Use the "RESET_CACHE Procedure" instead of this deprecated procedure.

To increase performance, Oracle Application Express caches the results of authorization schemes after they have been evaluated. You can use this procedure to undo caching, requiring each authorization scheme be revalidated when it is next encountered during page show or accept processing. You can use this procedure if
you want users to have the ability to change their responsibilities (their authorization profile) within your application.

Syntax

APEX_UTIL.RESET_AUTHORIZATIONS;

Parameters

None.

Example

The following example shows how to use the RESET_AUTHORIZATIONS procedure to clear the authorization scheme cache.

BEGIN
    APEX_UTIL.RESET_AUTHORIZATIONS;
END;

36.101 RESET_PASSWORD Procedure

This procedure is used to change the password of a given user name for the current workspace. This procedure changes the password of p_user_name in the current workspace to p_new_password. If p_change_password_on_first_use is TRUE, then the user has to change the password on the next login.

Syntax

APEX_UTIL.RESET_PASSWORD (  
    p_user_name      IN VARCHAR2 DEFAULT WWW_FLOW_SECURITY.G_USER,  
    p_old_password   IN VARCHAR2 DEFAULT NULL,  
    p_new_password   IN VARCHAR2,  
    p_change_password_on_first_use IN BOOLEAN DEFAULT TRUE );

Parameters

Table 36-84  RESET_PASSWORD Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user whose password should be changed. The default is the currently logged in Application Express user name.</td>
</tr>
<tr>
<td>p_old_password</td>
<td>The current password of the user. The call succeeds if the given value matches the current password or it is null and the owner of the calling PL/SQL code has APEX_ADMINISTRATOR_ROLE. If the value is not the user's password, an error occurs.</td>
</tr>
<tr>
<td>p_new_password</td>
<td>The new password.</td>
</tr>
</tbody>
</table>
Table 36-84  (Cont.) RESET_PASSWORD Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_change_password_on_first_use</td>
<td>If TRUE (default), the user must change the password on the next login.</td>
</tr>
</tbody>
</table>

Error Returns

Table 36-85  RESET_PASSWORD Parameters

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_CREDENTIALS</td>
<td>Occurs if p_user_name does not match p_old_password,</td>
</tr>
<tr>
<td>APEX.AUTHENTICATION.LOGIN_THROTTL E.COUNTER</td>
<td>Indicates authentication prevented by login throttle.</td>
</tr>
<tr>
<td>internal error</td>
<td>Occurs if p_old_password is NULL and caller does not have APEX_ADMINISTRATOR_ROLE.</td>
</tr>
<tr>
<td>internal error</td>
<td>Indicates caller is not a valid workspace schema.</td>
</tr>
</tbody>
</table>

Example

This example demonstrates changing the password of the currently logged in user to a new password.

```sql
apex_util.reset_password (  
    p_old_password => :P111_OLD_PASSWORD,  
    p_new_password => :P111_NEW_PASSWORD );
```

36.102 RESET_PW Procedure

This procedure resets the password for a named user and emails it in a message to the email address located for the named account in the current workspace. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```sql
APEX_UTIL.RESET_PW(  
    p_user IN VARCHAR2,  
    p_msg  IN VARCHAR2);  
```

Parameters

Table 36-86  RESET_PW Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>
Table 36-86  (Cont.) RESET_PW Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_msg</td>
<td>Message text to be mailed to a user.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the \texttt{RESET_PW} procedure to reset the password for the user ‘FRANK’.

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

See Also:

"CHANGE_CURRENT_USER_PW Procedure"

36.103 SAVEKEY_NUM Function

This function sets a \texttt{package variable} (\texttt{apex_utilities.g_val_num}) so that it can be retrieved using the function \texttt{KEYVAL_NUM}.

Syntax

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

Parameters

Table 36-87  SAVEKEY_NUM Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_val</td>
<td>The numeric value to be saved.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the \texttt{SAVEKEY_NUM} function to set the \texttt{apex_utilities.g_val_num} \texttt{package variable} to the value of 10.

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

See Also:
"KEYVAL_NUM Function"

36.104 SAVEKEY_VC2 Function

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

Syntax

APEX_UTIL.SAVEKEY_VC2(
    p_val IN VARCHAR2
) RETURN VARCHAR2;

Parameters

Table 36-88 SAVEKEY_VC2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_val</td>
<td>The is the VARCHAR2 value to be saved.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SAVEKEY_VC2 function to set the apex_utilities.g_val_vc2 package variable to the value of 'XXX'.

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

See Also:
"KEYVAL_VC2 Function"

36.105 SET_APP_BUILD_STATUS Procedure

This procedure sets the build status of the specified application.
Syntax

apex_util.set_app_build_status( p_application_id IN NUMBER,
            p_build_status in VARCHAR2 );

Parameters

Table 36-89  SET_APP_BUILD_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application.</td>
</tr>
<tr>
<td>p_build_status</td>
<td>The new build status of the application. Values include:</td>
</tr>
<tr>
<td></td>
<td>• RUN_ONLY - The application can be run but cannot be edited by developers.</td>
</tr>
<tr>
<td></td>
<td>• RUN_AND_BUILD - The application can be run and can also be edited by developers.</td>
</tr>
</tbody>
</table>

Example

begin
    apex_util.set_app_build_status(
        p_application_id => 170,
        p_build_status   => 'RUN_ONLY' );
    commit;
end;

36.106 SET_APPLICATION_STATUS Procedure

This procedure changes the status of the application.

Syntax

APEX_UTIL.SET_APPLICATION_STATUS(
    p_application_id IN NUMBER,
    p_application_status IN VARCHAR2,
    p_unavailable_value IN VARCHAR2,
    p_restricted_user_list IN VARCHAR2);

Parameters

Table 36-90  SET_APPLICATION_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The Application ID.</td>
</tr>
</tbody>
</table>
### Table 36-90  (Cont.) SET_APPLICATION_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_status</td>
<td>New application status. Values include:</td>
</tr>
<tr>
<td></td>
<td>• AVAILABLE - Application is available with no restrictions.</td>
</tr>
<tr>
<td></td>
<td>• AVAILABLE_W_EDIT_LINK - Application is available with no restrictions.</td>
</tr>
<tr>
<td></td>
<td>Developer Toolbar shown to developers.</td>
</tr>
<tr>
<td></td>
<td>• DEVELOPERS_ONLY - Application only available to developers.</td>
</tr>
<tr>
<td></td>
<td>• RESTRICTED_ACCESS - Application only available to users in p_restricted_user_list.</td>
</tr>
<tr>
<td></td>
<td>• UNAVAILABLE - Application unavailable. Message shown in p_unavailable_value.</td>
</tr>
<tr>
<td></td>
<td>• UNAVAILABLE_PLSQL - Application unavailable. Message shown from PL/SQL block in p_unavailable_value.</td>
</tr>
<tr>
<td></td>
<td>• UNAVAILABLE_URL - Application unavailable. Redirected to URL provided in p_unavailable_value.</td>
</tr>
<tr>
<td>p_unavailable_value</td>
<td>Value used when application is unavailable. This value has different semantics dependent upon value for p_application_status.</td>
</tr>
<tr>
<td>p_restricted_user_list</td>
<td>Comma separated list of users permitted to access application, when p_application_status = RESTRICTED_ACCESS.</td>
</tr>
</tbody>
</table>

### Examples

```sql
begin
  apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'AVAILABLE'
  );
end;

begin
  apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'AVAILABLE_W_EDIT_LINK'
  );
end;

begin
  apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'DEVELOPERS_ONLY'
  );
end;

begin
  apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'RESTRICTED_ACCESS',
    p_restricted_user_list => 'xxx.xxx@abc.com'
  );
```
begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE',
    p_unavailable_value => 'Application not available, sorry ');
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE_PLSQL',
    p_unavailable_value => 'sys.htp.p('Application unavailable, sorry');' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE_URL',
    p_unavailable_value => 'http://www.xyz.com' );
end;

See Also:
"Availability" in Oracle Application Express App Builder User's Guide

36.107 SET_ATTRIBUTE Procedure

This procedure sets the value of one of the attribute values (1 through 10) of a user in the Application Express accounts table.

Syntax

APEX_UTIL.SET_ATTRIBUTE(  
    p_userid IN NUMBER,  
    p_attribute_number IN NUMBER,  
    p_attribute_value IN VARCHAR2);

Parameters

Table 36-91   SET_ATTRIBUTE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_userid</td>
<td>The numeric ID of the user account.</td>
</tr>
<tr>
<td>p_attribute_number</td>
<td>Attribute number in the user record (1 through 10).</td>
</tr>
</tbody>
</table>
Table 36-91  (Cont.) SET_ATTRIBUTE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_attribute_value</td>
<td>Value of the attribute located by p_attribute_number to be set in the user record.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_ATTRIBUTE procedure to set the number 1 attribute for user ‘FRANK’ with the value ‘foo’.

DECLARE
    VAL VARCHAR2(4000);
BEGIN
    APEX_UTIL.SET_ATTRIBUTE (p_userid => apex_util.get_user_id(p_username => 'FRANK'),
        p_attribute_number => 1,
        p_attribute_value => 'foo');
END;

See Also:

"GET_ATTRIBUTE Function"

36.108 SET_AUTHENTICATION_RESULT Procedure

This procedure can be called from an application’s custom authentication function (that is, credentials verification function). The status passed to this procedure is logged in the Login Access Log.

Syntax

APEX_UTIL.SET_AUTHENTICATION_RESULT(p_code IN NUMBER);

Parameters

Table 36-92  SET_AUTHENTICATION_RESULT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_code</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

```sql
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2
) RETURN BOOLEAN IS BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is back.);
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

See Also:

- "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*
- "GET_AUTHENTICATION_RESULT Function"
- "SET_CUSTOM_AUTH_STATUS Procedure"

36.109 SET_BUILD_OPTION_STATUS Procedure

Use this procedure to change the build option status of a specified application.

Note:

The build option status will be overwritten when the application is upgraded to a new version. To keep the status set via the API, it is necessary to set the build option attribute `On Upgrade Keep Status` to Yes.
Syntax

apex_util.set_build_option_status(p_application_id IN NUMBER,
    p_id IN NUMBER,
    p_build_status IN VARCHAR2);

Parameters

Table 36-93    SET_BUILD_OPTION_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The ID of the application that owns the build option under shared components.</td>
</tr>
<tr>
<td>p_id</td>
<td>The ID of the build option in the application.</td>
</tr>
<tr>
<td>p_build_status</td>
<td>The new status of the build option. Possible values are INCLUDE, EXCLUDE.</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates how to use the `SET_BUILD_OPTION_STATUS` procedure to change the current status of build option.

```
BEGIN
APEX_UTIL.SET_BUILD_OPTION_STATUS(
    P_APPLICATION_ID => 101,
    P_ID => 245935500311121039, P_BUILD_STATUS=>'INCLUDE');
END;
```

36.110 SET_CURRENT_THEME_STYLE Procedure [DEPRECATED]

This procedure sets the user interface theme style for an application. For example, if there are more than one theme styles available for the current theme, you can use this procedure to change the application theme style.

Syntax

APEX_UTIL.SET_CURRENT_THEME_STYLE(
    p_theme_number IN NUMBER,
    p_theme_style_id IN NUMBER
);
Parameters

Table 36-94  SET_CURRENT_THEME_STYLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_theme_number</td>
<td>The current theme number of the application. This can be retrieved from APEX_APPLICATION_THEMES view.</td>
</tr>
<tr>
<td>p_theme_style_id</td>
<td>The numeric ID of theme style. You can get available theme styles for an application from APEX_APPLICATION_THEME_STYLES view.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_CURRENT_THEME_STYLE procedure to set the current application desktop theme style to Blue.

DECLARE
    l_current_theme_number number;
    l_theme_style_id       number;
BEGIN
    select theme_number
      into l_current_theme_number
    from apex_application_themes
    where application_id = :app_id
    and ui_type_name = 'DESKTOP'
    and is_current = 'Yes';

    select s.theme_style_id
      into l_new_theme_style_id
    from apex_application_theme_styles s, apex_application_themes t
    where s.application_id = t.application_id
    and s.theme_number = t.theme_number
    and s.application_id = :app_id
    and t.ui_type_name = 'DESKTOP'
    and t.is_current = 'Yes'
    and s.name = 'Blue';

    if l_current_theme_number is not null and
    l_new_theme_style_id is not null then
        APEX_UTIL.SET_CURRENT_THEME_STYLE(
            p_theme_number   => l_current_theme_number,
            p_theme_style_id => l_new_theme_style_id
        );
    end if;
END;
36.111 SET_CUSTOM_AUTH_STATUS Procedure

This procedure can be called from an application’s custom authentication function (that is, credentials verification function). The status passed to this procedure is logged in the Login Access Log.

Syntax

APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status IN VARCHAR2);

Parameters

Table 36-95 SET_CUSTOM_AUTH_STATUS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_status</td>
<td>Any text the developer chooses to denote the result of the authentication attempt (up to 4000 characters).</td>
</tr>
</tbody>
</table>

Example

One way to use the SET_CUSTOM_AUTH_STATUS procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. Note that no credentials verification is performed. The status set using this procedure is visible in the apex_user_access_log view and in the reports on this view available to workspace and site administrators.

CREATE OR REPLACE FUNCTION MY_AUTH(
  p_username IN VARCHAR2,
  p_password IN VARCHAR2)
RETURN BOOLEAN IS BEGIN
  APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is back.');
  IF UPPER(p_username) = 'GOOD' THEN
    APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
    RETURN TRUE;
  ELSE
    APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
    RETURN FALSE;
  END IF;
END;
See Also:

- "Monitoring Activity within a Workspace" in Oracle Application Express Administration Guide
- "SET_AUTHENTICATION_RESULT Procedure"
- "GET_AUTHENTICATION_RESULT Function"

36.112 SET_EDITION Procedure

This procedure sets the name of the edition to be used in all application SQL parsed in the current page view or page submission.

Syntax

```sql
APEX_UTIL.SET_EDITION(
    p_edition IN VARCHAR2);
```

Parameters

Table 36-96  SET_EDITION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

Example

The following example shows how to use the SET_EDITION procedure. It sets the edition name for the database session of the current page view.

```sql
BEGIN
    APEX_UTIL.SET_EDITION( P_EDITION => 'Edition1' );
END;
```

Note:

Support for Edition-Based Redefinition is only available in database version 11.2.0.1 or higher.

36.113 SET_EMAIL Procedure

This procedure updates a user account with a new email address. To execute this procedure, the current user must have administrative privileges in the workspace.
Syntax

APEX_UTIL.SET_EMAIL(
    p_userid IN NUMBER,
    p_email  IN VARCHAR2);

Parameters

Table 36-97  SET_EMAIL Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_userid</td>
<td>The numeric ID of the user account.</td>
</tr>
<tr>
<td>p_email</td>
<td>The email address to be saved in user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_EMAIL procedure to set the value of EMAIL to 'frank.scott@somewhere.com' for the user 'FRANK'.

BEGIN
    APEX_UTIL.SET_EMAIL(
        p_userid  => APEX_UTIL.GET_USER_ID('FRANK'),
        p_email   => 'frank.scott@somewhere.com');
END;

See Also:

- "GET_EMAIL Function"
- "GET_USER_ID Function"

36.114 SET_FIRST_NAME Procedure

This procedure updates a user account with a new FIRST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

APEX_UTIL.SET_FIRST_NAME(
    p_userid    IN NUMBER,
    p_first_name IN VARCHAR2);
Parameters

Table 36-98 SET_FIRST_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_userid</td>
<td>The numeric ID of the user account.</td>
</tr>
<tr>
<td>p_first_name</td>
<td>FIRST_NAME value to be saved in user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_FIRST_NAME procedure to set the value of FIRST_NAME to 'FRANK' for the user 'FRANK'.

BEGIN
    APEX_UTIL.SET_FIRST_NAME(
        p_userid       => APEX_UTIL.GET_USER_ID('FRANK'),
        p_first_name   => 'FRANK');
END;

See Also:

- "GET_FIRST_NAME Function"
- "GET_USER_ID Function"

36.115 SET_GLOBAL_NOTIFICATION Procedure

This procedure is used to set the global notification message which is the message displayed in page #GLOBAL_NOTIFICATION# substitution string.

Syntax

APEX_UTIL.SET_GLOBAL_NOTIFICATION(
    p_application_id IN NUMBER,
    p_global_notification_message IN VARCHAR2);

Parameters

Table 36-99 SET_GLOBAL_NOTIFICATION Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_application_id</td>
<td>The Application ID.</td>
</tr>
<tr>
<td>p_global_notification_message</td>
<td>Text string to be used for the global notification message.</td>
</tr>
</tbody>
</table>
Example

begin
    apex_util.set_global_notification(
        p_application_id => 117,
        p_global_notification_message => 'This application will be upgraded this weekend at 2100 UTC'
    );
end;

See Also:

"Availability" in Oracle Application Express App Builder User's Guide

36.116 SET_GROUP_GROUP_GRANTS Procedure

This procedure modifies the group grants for a given group.

Syntax

APEX_UTIL.SET_GROUP_GROUP_GRANTS (  
p_group_name IN VARCHAR2,  
p_granted_group_names IN apex_t_varchar2 );

Parameters

Table 36-100  SET_GROUP_GROUP_GRANTS Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_group_name</td>
<td>The target group name.</td>
</tr>
<tr>
<td>p_granted_group_names</td>
<td>The names of groups to grant to p_group_name.</td>
</tr>
</tbody>
</table>

Example

This example creates three groups (ACCTS_PAY, ACCTS_REC, MANAGER) and then grants ACCTS_PAY and ACCTS_REC to MANAGER.

apex_util.create_user_group (  
p_group_name => 'ACCTS_PAY' );

apex_util.create_user_group (  
p_group_name => 'ACCTS_REC' );

apex_util.create_user_group (  
p_group_name => 'MANAGER' );

apex_util.set_group_group_grants (  
p_group_name => 'MANAGER',  
p_granted_group_names => apex_t_varchar2('ACCTS_PAY', 'ACCTS_REC') );
36.117 SET_GROUP_USER_GRANTS Procedure

This procedure modifies the group grants for a given user.

Syntax

APEX_UTIL.SET_GROUP_USER_GRANTS (  
    p_user_name IN VARCHAR2,  
    p_granted_group_names IN apex_t_varchar2 );

Parameters

Table 36-101  SET_GROUP_USER_GRANTS Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The target user name.</td>
</tr>
<tr>
<td>p_granted_group_names</td>
<td>The names of groups to grant to p_user_name.</td>
</tr>
</tbody>
</table>

Example

This example creates a user group (MANAGER) and a user (Example User) and then grants MANAGER to Example User.

```plaintext
apex_util.create_user_group (  
    p_group_name => 'MANAGER' );
apex_util.create_user (  
    p_user_name => 'Example User',  
    p_web_password => 1_random_password );
-- grant MANAGER to Example User
apex_util.set_group_user_grants (  
    p_user_name => 'Example User',  
    p_granted_group_names => apex_t_varchar2('MANAGER') );
```

36.118 SET_LAST_NAME Procedure

This procedure updates a user account with a new LAST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

APEX_UTIL.SET_LAST_NAME(  
    p_userid      IN NUMBER,  
    p_last_name   IN VARCHAR2 );
Parameters

Table 36-102  SET_LAST_NAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_userid</td>
<td>The numeric ID of the user account.</td>
</tr>
<tr>
<td>p_last_name</td>
<td>LAST_NAME value to be saved in the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_LAST_NAME procedure to set the value of LAST_NAME to 'SMITH' for the user 'FRANK'.

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

See Also:

- "GET_LAST_NAME Function"
- "GET_USER_ID Function"

36.119 SET_PARSING_SCHEMA_FOR_REQUEST Procedure

This procedure changes the parsing user for the current page view to another workspace schema. You can call this procedure only from within the application's Initialization PL/SQL Code.

Syntax

```sql
procedure set_parsing_schema_for_request (  
    p_schema in varchar2 );
```

Parameters

Table 36-103  SET_PARSING_SCHEMA_FOR_REQUEST Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_schema</td>
<td>The new parsing schema.</td>
</tr>
</tbody>
</table>
 Raises

PROGRAM_ERROR when not called from Initialization PL/SQL Code.
WWV_FLOW.NO_PRIV_ON_SCHEMA if p_schema is not a valid workspace schema.

Example

On pages 1-100, change the parsing schema to :G_PARSING_SCHEMA.

if :APP_PAGE_ID between 1 and 100 then
    apex_util.set_parsing_schema_for_request (
        p_schema => :G_PARSING_SCHEMA );
end if;

36.120 SET_PREFERENCE Procedure

This procedure sets a preference that persists beyond the user's current session.

Syntax

APEX_UTIL.SET_PREFERENCE (
    p_preference   IN    VARCHAR2 DEFAULT NULL,
    p_value        IN    VARCHAR2 DEFAULT NULL,
    p_user         IN    VARCHAR2 DEFAULT NULL);

Parameters

Table 36-104  SET_PREFERENCE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_preference</td>
<td>Name of the preference (case-sensitive).</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of the preference.</td>
</tr>
<tr>
<td>p_user</td>
<td>User for whom the preference is being set.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_PREFERENCE procedure to set a preference called 'default_view' to the value 'WEEKLY' that persists beyond session for the currently authenticated user.

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

Use this procedure with `apex_util.find_security_group_id` to ease the use of the mail package in batch mode. This procedure is especially useful when a schema is associated with more than one workspace. For example, you might want to create a procedure that is run by a nightly job to email all outstanding tasks.

**Syntax**

```sql
APEX_UTIL.SET_SECURITY_GROUP_ID (p_security_group_id IN NUMBER);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_security_group_id</td>
<td>This is the security group id of the workspace you are working in.</td>
</tr>
</tbody>
</table>

**Example**

The following example sends an alert to each user that has had a task assigned within the last day.

```sql
create or replace procedure new_tasks
is
    l_workspace_id      number;
    l_subject           varchar2(2000);
    l_body              clob;
    l_body_html         clob;
begin
    l_workspace_id := apex_util.find_security_group_id (p_workspace => 'PROJECTS');
    apex_util.set_security_group_id (p_security_group_id => l_workspace_id);
    l_body := 'You have new tasks';
    l_subject := 'You have new tasks';
    for c1 in (select distinct(p.email_address) email_address, p.user_id
        from teamsp_user_profile p, teamsp_tasks t
        where p.user_id = t.assigned_to_user_id
        and t.created_on > sysdate - 1
        and p.email_address is not null) loop
        l_body_html := l_body_html || l_body;
    end loop;
end;
```

See Also:

- “GET_PREFERENCE Function”
- “REMOVE_PREFERENCE Procedure”
l_body_html := '<p>The following tasks have been added.';
for c2 in (select task_name, due_date
    from teamsp_tasks
    where assigned_to_user_id = c1.user_id
    and created_on > sysdate - 1 ) loop
    l_body_html := l_body_html || '<p>Task: ||c2.task_name||, due '||c2.due_date;
    end loop;
apex_mail.send (
    p_to        => c1.email_address,
    p_from      => c1.email_address,
    p_body      => l_body,
    p_body_html => l_body_html,
    p_subj      => l_subject );
end loop;
end;

36.122 SET_SESSION_HIGH_CONTRAST_OFF Procedure

This procedure switches off high contrast mode for the current session.

Syntax

APEX_UTIL.SET_SESSION_HIGH_CONTRAST_OFF;

Parameters

None.

Example

In this example, high contrast mode is switched off for the current session.

BEGIN
    apex_util.set_session_high_contrast_off;
END;

36.123 SET_SESSION_HIGH_CONTRAST_ON Procedure

This procedure switches on high contrast mode for the current session.

Syntax

APEX_UTIL.SET_SESSION_HIGH_CONTRAST_ON;

Parameters

None.
Example
In this example, the current session is put into high contrast mode.

BEGIN
  apex_util.set_session_high_contrast_on;
END;

36.124 SET_SESSION_LANG Procedure

This procedure sets the language to be used for the current user in the current Application Express session. The language must be a valid IANA language name.

Syntax

APEX_UTIL.SET_SESSION_LANG(
  p_lang IN VARCHAR2);

Parameters

Table 36-106   SET_SESSION_LANG Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_lang</td>
<td>This is an IANA language code. Some examples include: en, de, de-at, zh-cn, and pt-br.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_SESSION_LANG procedure. It sets the language for the current user for the duration of the Application Express session.

BEGIN
  APEX_UTIL.SET_SESSION_LANG( P_LANG => 'en');
END;

36.125 SET_SESSION_LIFETIME_SECONDS Procedure

This procedure sets the current session's Maximum Session Length in Seconds value, overriding the corresponding application attribute. This allows developers to dynamically shorten or lengthen the session life based on criteria determined after the user authenticates.

Syntax

APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(
  p_seconds IN NUMBER,
  p_scope IN VARCHAR2 DEFAULT 'SESSION');
Parameters

Table 36-107  SET_SESSION_LIFETIME_SECONDS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_seconds</td>
<td>A positive integer indicating the number of seconds the session used by this application is allowed to exist.</td>
</tr>
<tr>
<td>p_scope</td>
<td>This parameter is obsolete. The procedure always sets the lifetime for the whole session.</td>
</tr>
</tbody>
</table>

Example 1

The following example shows how to use the SET_SESSION_LIFETIME_SECONDS procedure to set the current application's Maximum Session Length in Seconds attribute to 7200 seconds (two hours).

By allowing the p_scope input parameter to use the default value of 'SESSION', the following example would actually apply to all applications using the current session. This would be the most common use case when multiple Application Express applications use a common authentication scheme and are designed to operate as a suite in a common session.

BEGIN
    APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(p_seconds => 7200);
END;

Example 2

The following example shows how to use the SET_SESSION_LIFETIME_SECONDS procedure to set the current application's Maximum Session Length in Seconds attribute to 3600 seconds (one hour).

BEGIN
    APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(p_seconds => 3600);
END;

36.126 SET_SESSION_MAX_IDLE_SECONDS Procedure

Sets the current application's Maximum Session Idle Time in Seconds value for the current session, overriding the corresponding application attribute. This allows developers to dynamically shorten or lengthen the maximum idle time allowed between page requests based on criteria determined after the user authenticates.

Syntax

APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS (  
    p_seconds  IN    NUMBER,  
    p_scope    IN    VARCHAR2 DEFAULT 'SESSION');
Parameters

Table 36-108  SET_SESSION_MAX_IDLE_SECONDS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_seconds</td>
<td>A positive integer indicating the number of seconds allowed between page requests.</td>
</tr>
<tr>
<td>p_scope</td>
<td>This parameter is obsolete. The procedure always sets the lifetime for the whole session</td>
</tr>
</tbody>
</table>

Example 1

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 1200 seconds (twenty minutes). The following example applies to all applications using the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 1200);
END;
```

Example 2

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 600 seconds (ten minutes). This example applies to all applications using the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 600);
END;
```

36.127 SET_SESSION_SCREEN_READER_OFF Procedure

This procedure switches off screen reader mode for the current session.

Syntax

```
APEX_UTIL.SET_SESSION_SCREEN_READER_OFF;
```

Parameters

None
Example
In this example, the current session is put into standard mode.

BEGIN
    apex_util.set_session_screen_reader_off;
END;

36.129 SET_SESSION_STATE Procedure
This procedure sets session state for a current Oracle Application Express session.

Syntax
APEX_UTIL.SET_SESSION_STATE ( 
    p_name     IN    VARCHAR2 DEFAULT NULL,
    p_value    IN    VARCHAR2 DEFAULT NULL,
    p_commit   IN    BOOLEAN  DEFAULT TRUE);

Parameters
Table 36-109    SET_SESSION_STATE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_name</td>
<td>Name of the application-level or page-level item for which you are setting sessions state.</td>
</tr>
<tr>
<td>p_value</td>
<td>Value of session state to set.</td>
</tr>
</tbody>
</table>
Table 36-109   (Cont.) SET_SESSION_STATE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_commit</td>
<td>If true (the default), commit after modifying session state. If false or if the existing value in session state equals p_value, no commit is issued.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_SESSION_STATE procedure to set the value of the item 'my_item' to 'myvalue' in the current session.

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

See Also:

- "GET_SESSION_STATE Function"
- "GET_NUMERIC_SESSION_STATE Function"
- "Understanding Session State Management" in Oracle Application Express App Builder User's Guide

36.130 SET_SESSION_TERRITORY Procedure

This procedure sets the territory to be used for the current user in the current Application Express session. The territory name must be a valid Oracle territory.

Syntax

APEX_UTIL.SET_SESSION_TERRITORY(
    p_territory IN VARCHAR2);

Parameters

Table 36-110   SET_SESSION_TERRITORY Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_territory</td>
<td>A valid Oracle territory name. Examples include: AMERICA, UNITED KINGDOM, ISRAEL, AUSTRIA, and UNITED ARAB EMIRATES.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the SET_SESSION_TERRITORY procedure. It sets the territory for the current user for the duration of the Application Express session.

BEGIN
    APEX_UTIL.SET_SESSION_TERRITORY( P_TERRITORY => 'UNITED KINGDOM');
END;

36.131 SET_SESSION_TIME_ZONE Procedure

This procedure sets the time zone to be used for the current user in the current Application Express session.

Syntax

APEX_UTIL.SET_SESSION_TIME_ZONE(
    p_time_zone IN VARCHAR2);

Parameters

Table 36-11 SET_SESSION_TIME_ZONE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_timezone</td>
<td>A time zone value in the form of hours and minutes. Examples include: +09:00, 04:00, -05:00.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_SESSION_TIME_ZONE procedure. It sets the time zone for the current user for the duration of the Application Express session.

BEGIN
    APEX_UTIL.SET_SESSION_TIME_ZONE( P_TIME_ZONE => '-05:00');
END;

36.132 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 36-112  SET_USERNAME Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_userid</td>
<td>The numeric ID of the user account.</td>
</tr>
<tr>
<td>p_username</td>
<td>USER_NAME value to be saved in the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the SET_USERNAME procedure to set the value of USER_NAME to 'USER-XRAY' for the user 'FRANK'.

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

See Also:

- "GET_USERNAME Function"
- "GET_USER_ID Function"

36.133 SET_WORKSPACE_Procedure

This procedure sets the current workspace.

Syntax

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

Parameters

Table 36-113 describes the parameters available in SET_WORKSPACE_Procedure.

Table 36-113  SET_WORKSPACE_Procedure Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_workspace</td>
<td>The workspace's short name.</td>
</tr>
</tbody>
</table>
Example

This example shows how to set workspace `MY_WORKSPACE`.

```apex
apex_util.set_workspace {
    p_workspace => 'MY_WORKSPACE' );
```

36.134 SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure

This procedure displays a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches the high contrast mode on.

Syntax

```apex
APEX_UTIL.SHOW_HIGH_CONTRAST_MODE_TOGGLE ( 
    p_on_message  in varchar2 default null, 
    p_off_message in varchar2 default null); 
```

Parameters

Table 36-114  SHOW_HIGH_CONTRAST_MODE_TOGGLE Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_on_message</td>
<td>Optional text used for the link to switch to high contrast mode, when you are in standard mode. If this parameter is not passed, the default 'Set High Contrast Mode On' text is displayed.</td>
</tr>
<tr>
<td>p_off_message</td>
<td>Optional text used for the link to switch to standard mode, when you are in high contrast mode. If this parameter is not passed, the default 'Set High Contrast Mode Off' text is displayed.</td>
</tr>
</tbody>
</table>

Example

When running in standard mode, this procedure displays a link, Set High Contrast Mode On, that when clicked refreshes the current page and switches on high contrast mode. When running in high contrast mode, a link, Set High Contrast Mode Off, is displayed, that refreshes the current page and switches back to standard mode when clicked.

```apex
BEGIN
    apex_util.show_high_contrast_mode_toggle;
END;
```
Note:

There are also 2 translatable system messages that can be overridden at application level to change the default link text that is returned for this toggle. They include:

- `APEX.SET_HIGH_CONTRAST_MODE_OFF` - Default text = Set High Contrast Mode Off
- `APEX.SET_HIGH_CONTRAST_MODE_ON` - Default text = Set High Contrast Mode On

See Also:

"GET_HIGH_CONTRAST_MODE_TOGGLE Function"

36.135 SHOW_SCREEN_READER_MODE_TOGGLE Procedure

This procedure displays a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches the screen reader mode on.

Syntax

```
APEX_UTIL.SHOW_SCREEN_READER_MODE_TOGGLE (  
    p_on_message  IN VARCHAR2 DEFAULT NULL,  
    p_off_message IN VARCHAR2 DEFAULT NULL)
```

Parameters

Table 36-115  SHOW_SCREEN_READER_MODE_TOGGLE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_on_message</code></td>
<td>Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text is displayed.</td>
</tr>
<tr>
<td><code>p_off_message</code></td>
<td>Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text is displayed.</td>
</tr>
</tbody>
</table>

Example

When running in standard mode, this procedure displays a link 'Set Screen Reader Mode On', that when clicked refreshes the current page and switches on screen reader mode. When running in screen reader mode, a link 'Set Screen Reader Mode
Off' is displayed, that when clicked refreshes the current page and switches back to standard mode.

```
BEGIN
    apex_util.show_screen_reader_mode_toggle;
END;
```

### 36.136 STRING_TO_TABLE Function [DEPRECATED]

Oracle recommends that you use the `SPLIT` and `SPLIT_NUMBERS` functions.

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 36-116 STRING_TO_TABLE Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>String to be converted into a PL/SQL table of type <code>APEX_APPLICATION_GLOBAL.VC_ARR2</code>.</td>
</tr>
<tr>
<td>p_separator</td>
<td>String separator. The default is a colon.</td>
</tr>
</tbody>
</table>

**Example**

The following example shows how to use the `STRING_TO_TABLE` function. The function is passed the string 'One:Two:Three' in the `p_string` parameter and it returns a PL/SQL array of type `APEX_APPLICATION_GLOBAL.VC_ARR2` containing 3 elements, the element at position 1 contains the value 'One', position 2 contains the value 'Two' and position 3 contains the value 'Three'. This is then output using the `HTP.P` function call.

```
DECLARE
    l_vc_arr2    APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_vc_arr2 := APEX_UTIL.STRING_TO_TABLE('One:Two:Three');
    FOR z IN 1..l_vc_arr2.count LOOP
        htp.p(l_vc_arr2(z));
    END LOOP;
END;
```
36.137 STRONG_PASSWORD_CHECK Procedure

This procedure returns Boolean OUT values based on whether a proposed password meets the password strength requirements as defined by the Oracle Application Express site administrator.

Syntax

APEX_UTIL.STRONG_PASSWORD_CHECK(
  p_username                    IN  VARCHAR2,
  p_password                    IN  VARCHAR2,
  p_old_password                IN  VARCHAR2,
  p_workspace_name              IN  VARCHAR2,
  p_use_strong_rules            IN  BOOLEAN,
  p_min_length_err              OUT BOOLEAN,
  p_new_differs_by_err          OUT BOOLEAN,
  p_one_alpha_err               OUT BOOLEAN,
  p_one_numeric_err             OUT BOOLEAN,
  p_one_punctuation_err         OUT BOOLEAN,
  p_one_upper_err               OUT BOOLEAN,
  p_one_lower_err               OUT BOOLEAN,
  p_not_like_username_err       OUT BOOLEAN,
  p_not_like_workspace_name_err OUT BOOLEAN,
  p_not_like_words_err          OUT BOOLEAN,
  p_not_reusable_err            OUT BOOLEAN);

Parameters

Table 36-117  STRONG_PASSWORD_CHECK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Username that identifies the account in the current workspace.</td>
</tr>
<tr>
<td>p_password</td>
<td>Password to be checked against password strength rules.</td>
</tr>
<tr>
<td>p_old_password</td>
<td>Current password for the account. Used only to enforce &quot;new password must differ from old&quot; rule.</td>
</tr>
<tr>
<td>p_workspace_name</td>
<td>Current workspace name, used only to enforce &quot;password must not contain workspace name&quot; rule.</td>
</tr>
<tr>
<td>p_use_strong_rules</td>
<td>Pass FALSE when calling this API.</td>
</tr>
</tbody>
</table>
Table 36-117  (Cont.) STRONG_PASSWORD_CHECK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_min_length_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets minimum length requirement.</td>
</tr>
<tr>
<td>p_new_differs_by_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets &quot;new password must differ from old&quot; requirements.</td>
</tr>
<tr>
<td>p_one_alpha_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets requirement to contain at least one alphabetic character.</td>
</tr>
<tr>
<td>p_one_numeric_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one numeric character.</td>
</tr>
<tr>
<td>p_one_punctuation_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one punctuation character.</td>
</tr>
<tr>
<td>p_one_upper_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one upper-case character.</td>
</tr>
<tr>
<td>p_one_lower_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one lower-case character.</td>
</tr>
<tr>
<td>p_not_like_username_err</td>
<td>Result returns TRUE or FALSE depending upon whether the password meets requirements that it not contain the username.</td>
</tr>
<tr>
<td>p_not_like_workspace_name_err</td>
<td>Result returns TRUE or FALSE whether upon whether the password meets requirements that it not contain the workspace name.</td>
</tr>
<tr>
<td>p_not_like_words_err</td>
<td>Result returns TRUE or FALSE whether the password meets requirements that it not contain specified simple words.</td>
</tr>
<tr>
<td>p_not_reusable_err</td>
<td>Result returns TRUE or FALSE whether the password can be reused based on password history rules.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the STRONG_PASSWORD_CHECK procedure. It checks the new password 'foo' for the user 'SOMEBODY' meets all the password strength requirements defined by the Oracle Application Express site administrator. If any of the checks fail (the associated OUT parameter returns TRUE), then the example outputs a relevant message. For example, if the Oracle Application Express site administrator has defined that passwords must have at least one numeric character and the password 'foo' was checked, then the p_one_numeric_err OUT parameter would return TRUE and the message 'Password must contain at least one numeric character' would be output.

```
DECLARE
    l_username            varchar2(30);
    l_password            varchar2(30);
    l_old_password        varchar2(30);
    l_workspace_name      varchar2(30);
    l_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';
    l_password_history_days :=
        apex_instance_admin.get_parameter ('PASSWORD_HISTORY_DAYS');

    APEX_UTIL.STRONG_PASSWORD_CHECK(
        p_username => l_username,
        p_password => l_password,
        p_old_password => l_old_password,
        p_workspace_name => l_workspace_name,
        p_use_strong_rules => false,
        p_min_length_err => l_min_length_err,
        p_new_differs_by_err => l_new_differs_by_err,
        p_one_alpha_err => l_one_alpha_err,
        p_one_numeric_err => l_one_numeric_err,
        p_one_punctuation_err => l_one_punctuation_err,
        p_one_upper_err => l_one_upper_err,
        p_one_lower_err => l_one_lower_err,
        p_not_like_username_err => l_not_like_username_err,
        p_not_like_workspace_name_err => l_not_like_workspace_name_err,
        p_not_like_words_err => l_not_like_words_err,
        p_not_reusable_err => l_not_reusable_err);

    IF l_min_length_err THEN
        htp.p('Password is too short');
    END IF;

    IF l_new_differs_by_err THEN
        htp.p('Password is too similar to the old password');
    END IF;

    IF l_one_alpha_err THEN
        htp.p('Password must contain at least one alphabetic character');
    END IF;

    IF l_one_numeric_err THEN
        htp.p('Password must contain at least one numeric character');
    END IF;

    IF l_one_punctuation_err THEN
htp.p('Password must contain at least one punctuation character');
END IF;

IF l_one_upper_err THEN
  htp.p('Password must contain at least one upper-case character');
END IF;

IF l_one_lower_err THEN
  htp.p('Password must contain at least one lower-case character');
END IF;

IF l_not_like_username_err THEN
  htp.p('Password may not contain the username');
END IF;

IF l_not_like_workspace_name_err THEN
  htp.p('Password may not contain the workspace name');
END IF;

IF l_not_like_words_err THEN
  htp.p('Password contains one or more prohibited common words');
END IF;

IF l_not_reusable_err THEN
  htp.p('Password cannot be used because it has been used for the account within the last '||l_password_history_days||' days.');
END IF;
END;

See Also:

"Creating Strong Password Policies" in Oracle Application Express Administration Guide.

36.138 STRONG_PASSWORD_VALIDATION Function

This function returns formatted HTML in a VARCHAR2 result based on whether a proposed password meets the password strength requirements as defined by the Oracle Application Express site administrator.

Syntax

FUNCTION STRONG_PASSWORD_VALIDATION(
  p_username                    IN  VARCHAR2,
  p_password                    IN  VARCHAR2,
  p_old_password                IN  VARCHAR2 DEFAULT NULL,
  p_workspace_name              IN  VARCHAR2)
RETURN VARCHAR2;
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_username</td>
<td>Username that identifies the account in the current workspace.</td>
</tr>
<tr>
<td>p_password</td>
<td>Password to be checked against password strength rules.</td>
</tr>
<tr>
<td>p_old_password</td>
<td>Current password for the account. Used only to enforce &quot;new password must differ from old&quot; rule.</td>
</tr>
<tr>
<td>p_workspace_name</td>
<td>Current workspace name, used only to enforce &quot;password must not contain workspace name&quot; rule.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the STRONG_PASSWORD_VALIDATION procedure. It checks the new password 'foo' for the user 'SOMEBODY' meets all the password strength requirements defined by the Oracle Application Express site administrator. If any of the checks fail, then the example outputs formatted HTML showing details of where the new password fails to meet requirements.

```sql
DECLARE
  l_username        varchar2(30);
  l_password        varchar2(30);
  l_old_password    varchar2(30);
  l_workspace_name  varchar2(30);
BEGIN
  l_username := 'SOMEBODY';
  l_password  := 'foo';
  l_old_password := 'foo';
  l_workspace_name := 'XYX_WS';

  HTP.P(APEX_UTIL.STRONG_PASSWORD_VALIDATION(
    p_username       => l_username,
    p_password       => l_password,
    p_old_password   => l_old_password,
    p_workspace_name => l_workspace_name));
END;
```

36.139 SUBMIT_FEEDBACK Procedure

This procedure enables you to write a procedure to submit feedback, rather than using the page that can be generated by create page of type feedback.

Syntax

APEX_UTIL.SUBMIT_FEEDBACK (  
  p_comment IN VARCHAR2 DEFAULT NULL,  
  p_type IN NUMBER DEFAULT '1',  
  p_application_id IN VARCHAR2 DEFAULT NULL,  
)
Table 36-119  SUBMIT_FEEDBACK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_comment</td>
<td>Comment to be submitted.</td>
</tr>
<tr>
<td>p_type</td>
<td>Type of feedback (1 is General Comment, 2 is Enhancement Request, 3 is Bug).</td>
</tr>
<tr>
<td>p_application_id</td>
<td>ID of application related to the feedback.</td>
</tr>
<tr>
<td>p_page_id</td>
<td>ID of page related to the feedback.</td>
</tr>
<tr>
<td>p_email</td>
<td>Email of the user providing the feedback.</td>
</tr>
<tr>
<td>p_screen_width</td>
<td>Width of screen at time feedback was provided.</td>
</tr>
<tr>
<td>p_screen_height</td>
<td>Height of screen at time feedback was provided.</td>
</tr>
<tr>
<td>p_attribute_01</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_02</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_03</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_04</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_05</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_06</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_07</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_attribute_08</td>
<td>Custom attribute for collecting feedback.</td>
</tr>
<tr>
<td>p_label_01</td>
<td>Label for corresponding custom attribute.</td>
</tr>
<tr>
<td>p_label_02</td>
<td>Label for corresponding custom attribute.</td>
</tr>
<tr>
<td>p_label_03</td>
<td>Label for corresponding custom attribute.</td>
</tr>
<tr>
<td>p_label_04</td>
<td>Label for corresponding custom attribute.</td>
</tr>
</tbody>
</table>
Table 36-119  (Cont.) SUBMIT_FEEDBACK Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_label_05</td>
<td>Label for corresponding custom attribute.</td>
</tr>
<tr>
<td>p_label_06</td>
<td>Label for corresponding custom attribute.</td>
</tr>
<tr>
<td>p_label_07</td>
<td>Label for corresponding custom attribute.</td>
</tr>
<tr>
<td>p_label_08</td>
<td>Label for corresponding custom attribute.</td>
</tr>
</tbody>
</table>

Example

The following example submits a bug about page 22 within application 283.

```
begin
    apex_util.submit_feedback (
        p_comment => 'This page does not render properly for me',
        p_type => 3,
        p_application_id => 283,
        p_page_id => 22,
        p_email => 'user@xyz.corp',
        p_attribute_01 => 'Charting',
        p_label_01 => 'Component');
end;
/
```

36.140 SUBMIT_FEEDBACK_FOLLOWUP Procedure

This procedure enables you to submit follow up to a feedback.

Syntax

```
APEX_UTIL.SUBMIT_FEEDBACK_FOLLOWUP (
    p_feedback_id IN NUMBER,
    p_follow_up IN VARCHAR2 DEFAULT NULL,
    p_email IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 36-120  SUBMIT_FEEDBACK_FOLLOWUP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_feedback_followup</td>
<td>ID of feedback that this is a follow up to.</td>
</tr>
<tr>
<td>p_follow_up</td>
<td>Text of follow up.</td>
</tr>
<tr>
<td>p_email</td>
<td>Email of user providing the follow up.</td>
</tr>
</tbody>
</table>
Example

The following example submits follow up to a previously filed feedback.

```plsql
begin
    apex_util.submit_feedback_followup (
        p_feedback_id    => 12345,
        p_follow_up      => 'I tried this on another instance and it does not work there either',
        p_email          => 'user@xyz.corp');
end;
/
```

36.141 TABLE_TO_STRING Function [DEPRECATED]

Oracle recommends that you use the JOIN and JOIN_CLOB functions.

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**

```plsql
APEX_UTIL.TABLE_TO_STRING (
    p_table       IN     APEX_APPLICATION_GLOBAL.VC_ARR2,
    p_string      IN     VARCHAR2 DEFAULT ':')
RETURN VARCHAR2;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_string</td>
<td>String separator. Default separator is a colon (:).</td>
</tr>
<tr>
<td>p_table</td>
<td>PL/SQL table that is to be converted into a delimited string.</td>
</tr>
</tbody>
</table>

**Example**

The following function returns a comma delimited string of contact names that are associated with the provided cust_id.

```plsql
create or replace function get_contacts (
    p_cust_id  in  number
) return varchar2
is
    l_vc_arr2   apex_application_global.vc_arr2;
    l_contacts  varchar2(32000);
begin
    select contact_name
```
bulk collect
into l_vc_arr2
from contacts
where cust_id = p_cust_id
order by contact_name;

l_contacts := apex_util.table_to_string (
    p_table => l_vc_arr2,
    p_string => ', ');

return l_contacts;
end get_contacts;

See Also:

• “STRING_TO_TABLE Function [DEPRECATED]”
• “JOIN Function Signature 1”
• “JOIN Function Signature 2”
• “JOIN_CLOB Function”

36.142 UNEXPIRE_END_USER_ACCOUNT Procedure

Makes expired end users accounts and the associated passwords usable, enabling an end user to log in to developed applications.

Syntax

APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT (  
    p_user_name IN VARCHAR2);

Parameters

Table 36-122  UNEXPIRE_END_USER_ACCOUNT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

Example

The following example shows how to use the UNEXPIRE_END_USER_ACCOUNT procedure. Use this procedure to renew (unexpire) an Application Express end user account in the current workspace. This action specifically renews the account for use by end users to authenticate to developed applications and may also renew the account for use by developers or administrators to log in to a workspace.
This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```

**See Also:**
- "Table 36-23"
- "END_USER_ACCOUNT_DAYS_LEFT Function"

### 36.143 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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

**Example**

The following example shows how to use the `UNEXPIRE_WORKSPACE_ACCOUNT` procedure. Use this procedure to renew (unexpire) an Application Express workspace administrator account in the current workspace. This action specifically renews the account for use by developers or administrators to login to a workspace and may also renew the account for its use by end users to authenticate to developed applications.

This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.UNEXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```
36.144 UNLOCK_ACCOUNT Procedure

Sets a user account status to unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```sql
APEX_UTIL.UNLOCK_ACCOUNT(
    p_user_name IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>

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.

```sql
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        APEX_UTIL.UNLOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:' || c1.user_name || ' is now unlocked.');
    END LOOP;
END;
```
36.145 URL_ENCODE Function

The following special characters are encoded as follows:

<table>
<thead>
<tr>
<th>Special Characters</th>
<th>After Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>%25</td>
</tr>
<tr>
<td>+</td>
<td>%2B</td>
</tr>
<tr>
<td>space</td>
<td>+</td>
</tr>
<tr>
<td>.</td>
<td>%2E</td>
</tr>
<tr>
<td>*</td>
<td>%2A</td>
</tr>
<tr>
<td>?</td>
<td>%3F</td>
</tr>
<tr>
<td>\</td>
<td>%5C</td>
</tr>
<tr>
<td>/</td>
<td>%2F</td>
</tr>
<tr>
<td>&gt;</td>
<td>%3E</td>
</tr>
<tr>
<td>&lt;</td>
<td>%3C</td>
</tr>
<tr>
<td>)</td>
<td>%7B</td>
</tr>
<tr>
<td>(</td>
<td>%7D</td>
</tr>
<tr>
<td>~</td>
<td>%7E</td>
</tr>
<tr>
<td>[</td>
<td>%5B</td>
</tr>
<tr>
<td>]</td>
<td>%5D</td>
</tr>
<tr>
<td>'</td>
<td>%60</td>
</tr>
<tr>
<td>;</td>
<td>%3B</td>
</tr>
<tr>
<td>@</td>
<td>%40</td>
</tr>
<tr>
<td>&amp;</td>
<td>%26</td>
</tr>
<tr>
<td>#</td>
<td>%23</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>%5E</td>
</tr>
<tr>
<td>:</td>
<td>%3A</td>
</tr>
<tr>
<td>=</td>
<td>%3D</td>
</tr>
<tr>
<td>$</td>
<td>%24</td>
</tr>
</tbody>
</table>

Syntax

APEX_UTIL.URL_ENCODE (  
p_url IN VARCHAR2)  
RETURN VARCHAR2;
### Parameters

#### Table 36-125  URL_ENCODE Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The string to be encoded.</td>
</tr>
</tbody>
</table>

#### Example

The following example shows how to use the URL_ENCODE function.

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

In this example, the following URL:

http://www.myurl.com?id=1&cat=foo

Would be returned as:

http%3A%2F%2Fwww%2Emyurl%2Ecom%3Fid%3D1%26cat%3Dfoo

---

### 36.146 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

```sql
APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT (  
    p_user_name IN VARCHAR2)  
RETURN NUMBER;
```

#### Parameters

#### Table 36-126  WORKSPACE_ACCOUNT_DAYS_LEFT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_user_name</td>
<td>The user name of the user account.</td>
</tr>
</tbody>
</table>
Example

The following example shows how to use the WORKSPACE_ACCOUNT_DAYS_LEFT function. It can be used in to find the number of days remaining before an Application Express administrator or developer account in the current workspace expires.

DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from apex_users) LOOP
        l_days_left := APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT(p_user_name => c1.user_name);
        htp.p('Workspace Account:||c1.user_name|| expires in || l_days_left || days.');
    END LOOP;
END;

See Also:

- "EXPIRE_WORKSPACE_ACCOUNT Procedure"
- "UNEXPIRE_WORKSPACE_ACCOUNT Procedure"
APEX_WEB_SERVICE

The APEX_WEB_SERVICE API enables you to integrate other systems with Application Express by allowing you to interact with Web services anywhere you can use PL/SQL in your application. The API contains procedures and functions to call both SOAP and RESTful style Web services. It contains functions to parse the responses from Web services and to encode/decode into SOAP friendly base64 encoding. This API also contains package globals for managing cookies and HTTP headers when calling Web services whether from the API or by using standard processes of type Web service. Cookies and HTTP headers can be set before invoking a call to a Web service by populating the globals and the cookies and HTTP headers returned from the Web service response can be read from other globals.

- About the APEX_WEB_SERVICE API
- Invoking a SOAP Style Web Service
- Invoking a RESTful Style Web Service
- Retrieving Cookies and HTTP Headers
- Setting Cookies and HTTP Headers
- BLOB2CLOBBASE64 Function
- CLOBBASE642BLOB Function
- MAKE_REQUEST Procedure
- MAKE_REQUEST Function
- MAKE_REST_REQUEST Function
- MAKE_REST_REQUEST_B Function
- OAUTH_AUTHENTICATE Function
- OAUTH_GET_LAST_TOKEN Function
- OAUTH_SET_TOKEN Function
- PARSE_RESPONSE Function
- PARSE_RESPONSE_CLOB Function
- PARSE_XML Function
- PARSE_XML_CLOB Function

37.1 About the APEX_WEB_SERVICE API

Use the APEX_WEB_SERVICE API to invoke a Web service and examine the response anywhere you can use PL/SQL in Application Express.

The following are examples of when you might use the APEX_WEB_SERVICE API:

- When you want to invoke a Web service by using an On Demand Process using Ajax.
• When you want to invoke a Web service as part of an Authentication Scheme.
• When you need to pass a large binary parameter to a Web service that is base64 encoded.
• When you want to invoke a Web service as part of a validation.

37.2 Invoking a SOAP Style Web Service

There is a procedure and a function to invoke a SOAP style Web service. The procedure stores the response in the collection specified by the parameter p_collection_name. The function returns the results as an XMLTYPE. To retrieve a specific value from the response, you use either the PARSE_RESPONSE function if the result is stored in a collection or the PARSE_XML function if the response is returned as an XMLTYPE. To pass a binary parameter to the Web service as base64 encoded character data, use the function BLOB2CLOBBASE64. Conversely, to transform a response that contains a binary parameter that is base64 encoded use the function CLOBBASE642BLOB. The following is an example of using the BLOB2CLOBBASE64 function to encode a parameter, MAKE_REQUEST procedure to call a Web service, and the PARSE_RESPONSE function to extract a specific value from the response.

```
declare
    l_filename varchar2(255);
    l_BLOB BLOB;
    l_CLOB CLOB;
    l_envelope CLOB;
    l_response_msg varchar2(32767);
BEGIN
    IF :P1_FILE IS NOT NULL THEN
        SELECT filename, BLOB_CONTENT
        INTO l_filename, l_BLOB
        FROM APEX_APPLICATION_FILES
        WHERE name = :P1_FILE;

        l_CLOB := apex_web_service.blob2clobbase64(l_BLOB);

        l_envelope := q"<?xml version=1.0 encoding=UTF-8?>";
        <soapenv:Header/>
        <soapenv:Body>
            <chec:CheckInUniversal>
                <chec:dDocName>""||l_filename||""</chec:dDocName>
                <chec:dDocTitle>""||l_filename||""</chec:dDocTitle>
                <chec:dDocType>Document</chec:dDocType>
                <chec:dDocAuthor>GM</chec:dDocAuthor>
                <chec:dSecurityGroup>Public</chec:dSecurityGroup>
                <chec:dDocAccount></chec:dDocAccount>
                <chec:CustomDocMetaData>
                    <chec:property>
                        <chec:name>""</chec:name>
                        <chec:value>""</chec:value>
                    </chec:property>
            </chec:CheckInUniversal>
        </soapenv:Body>
    END IF;
```

Chapter 37
Invoking a SOAP Style Web Service

PARSEC_RESPONSE
37-2
37.3 Invoking a RESTful Style Web Service

RESTful style Web services use a simpler architecture than SOAP. Typically the input to a RESTful style Web service is a collection of name/value pairs. The response can be an XML document or simply text such as a comma separated response or JSON. The following is an example of MAKE_REST_REQUEST being used in an application process that is callable by Ajax.

```sql
declare
  l_clob clob;
  l_buffer         varchar2(32767);
  l_amount         number;
  l_offset         number;
begin
  l_clob := apex_web_service.make_rest_request(
    p_url => 'http://192.0.2.1/idc/idcplg',
    p_action => 'http://192.0.2.1/CheckIn/',
    p_collection_name => 'STELLENT_CHECKIN',
    p_envelope => l_envelope,
    p_username => 'sysadmin',
    p_password => 'password' );

  l_response_msg := apex_web_service.parse_response(
    p_collection_name=>'STELLENT_CHECKIN',
    p_xpath=>'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/idc:statusMessage/text()',
    p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');

  P1_RES_MSG := l_response_msg;
END IF;
END;
```
37.4 Retrieving Cookies and HTTP Headers

When you invoke a Web service using any of the supported methods in Application Express, the g_response_cookies and g_headers globals are populated if the Web service response included any cookies or HTTP headers. You can interrogate these globals and store the information in collections. The following are examples of interrogating the APEX_WEB_SERVICE globals to store cookie and HTTP header responses in collections.

```
declare
  i number;
  secure varchar2(1);
begin
  apex_collection.create_or_truncate_collection('P31_RESP_COOKIES');
  for i in 1.. apex_web_service.g_response_cookies.count loop
    IF (apex_web_service.g_response_cookies(i).secure) THEN
      secure := 'Y';
    ELSE
      secure := 'N';
    END IF;
    apex_collection.add_member(p_collection_name => 'P31_RESP_COOKIES',
                               p_c001 => apex_web_service.g_response_cookies(i).name,
                               p_c002 => apex_web_service.g_response_cookies(i).value,
                               p_c003 => apex_web_service.g_response_cookies(i).domain,
                               p_c004 => apex_web_service.g_response_cookies(i).expire,
                               p_c005 => apex_web_service.g_response_cookies(i).path,
                               p_c006 => secure,
                               p_c007 => apex_web_service.g_response_cookies(i).version );
  end loop;
end;
```
declare
    i number;
begin
    apex_collection.create_or_truncate_collection('P31_RESP_HEADERS');

    for i in 1.. apex_web_service.g_headers.count loop
        apex_collection.add_member(p_collection_name => 'P31_RESP_HEADERS',
            p_c001 => apex_web_service.g_headers(i).name,
            p_c002 => apex_web_service.g_headers(i).value,
            p_c003 => apex_web_service.g_status_code);
    end loop;
end;

37.5 Setting Cookies and HTTP Headers

You set cookies and HTTP headers that should be sent along with a Web service request by populating the globals \texttt{g\_request\_cookies} and \texttt{g\_request\_headers} before the process that invokes the Web service. The following examples show populating the globals to send cookies and HTTP headers with a request.

for c1 in (select seq_id, c001, c002, c003, c004, c005, c006, c007
    from apex_collections
        where collection_name = 'P31\_RESP\_COOKIES' ) loop
    apex_web_service.g_request_cookies(c1.seq_id).name := c1.c001;
    apex_web_service.g_request_cookies(c1.seq_id).value := c1.c002;
    apex_web_service.g_request_cookies(c1.seq_id).domain := c1.c003;
    apex_web_service.g_request_cookies(c1.seq_id).expire := c1.c004;
    apex_web_service.g_request_cookies(c1.seq_id).path := c1.c005;
    if c1.c006 = 'Y' then
        apex_web_service.g_request_cookies(c1.seq_id).secure := true;
    else
        apex_web_service.g_request_cookies(c1.seq_id).secure := false;
    end if;
    apex_web_service.g_request_cookies(c1.seq_id).version := c1.c007;
end loop;

for c1 in (select seq_id, c001, c002
    from apex_collections
        where collection_name = 'P31\_RESP\_HEADERS' ) loop
    apex_web_service.g_request_headers(c1.seq_id).name := c1.c001;
    apex_web_service.g_request_headers(c1.seq_id).value := c1.c002;
end loop;

37.6 BLOB2CLOBBASE64 Function

Use this function to convert a BLOB datatype into a CLOB that is base64 encoded. This is often used when sending a binary as an input to a Web service.
Syntax

APEX_WEB_SERVICE.BLOB2CLOBBASE64 (p_blob IN BLOB)
RETURN CLOB;

Parameters

Table 37-1  BLOB2CLOBBASE64 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_blob</td>
<td>The BLOB to convert into base64 encoded CLOB.</td>
</tr>
</tbody>
</table>

Example

The following example gets a file that was uploaded from the apex_application_files view and converts the BLOB into a CLOB that is base64 encoded.

```
DECLARE
    l_clob CLOB;
    l_blob BLOB;
BEGIN
    SELECT blob_content
    INTO l_blob
    FROM APEX_APPLICATION_FILES
    WHERE name = :P1_FILE;

    l_clob := apex_web_service.blob2clobbase64(l_blob);
END;
```

37.7 CLOBBASE642BLOB Function

Use this function to convert a CLOB datatype that is base64 encoded into a BLOB. This is often used when receiving output from a Web service that contains a binary parameter.

Syntax

APEX_WEB_SERVICE.CLOBBASE642BLOB (p_clob IN CLOB)
RETURN BLOB;

Parameters
Table 37-2   CLOBBASE642BLOB Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_clob</td>
<td>The base64 encoded CLOB to convert into a BLOB.</td>
</tr>
</tbody>
</table>

Example

The following example retrieves a base64 encoded node from an XML document as a CLOB and converts it into a BLOB.

```sql
declare
    l_base64    CLOB;
    l_blob     BLOB;
    l_xml       XMLTYPE;
begin
    l_base64 := apex_web_service.parse_xml_clob(l_xml, '/runReportReturn/reportBytes/text()');
    l_blob := apex_web_service.clobbase642blob(l_base64);
end;
```

37.8 MAKE_REQUEST Procedure

Use this procedure to invoke a SOAP style Web service with the supplied SOAP envelope and store the results in a collection.

Syntax

```sql
APEX_WEB_SERVICE.MAKE_REQUEST ( p_url               IN VARCHAR2,  p_action            IN VARCHAR2 default null,  p_version           IN VARCHAR2 default '1.1',  p_collection_name   IN VARCHAR2 default null,  p_envelope          IN CLOB,  p_username          IN VARCHAR2 default null,  p_password          IN VARCHAR2 default null,  p_scheme            IN VARCHAR2 DEFAULT 'Basic',  p_proxy_override    IN VARCHAR2 default null,  p_transfer_timeout  IN NUMBER   default 180,  p_wallet_path       IN VARCHAR2 default null,  p_wallet_pwd        IN VARCHAR2 default null,  p_https_host        IN VARCHAR2 default null );
```

Parameters

Table 37-3 describes the parameters available in the MAKE_REQUEST procedure.

Table 37-3   MAKE_REQUEST Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The URL endpoint of the Web service.</td>
</tr>
</tbody>
</table>
Table 37-3 (Cont.) MAKE_REQUEST Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_action</td>
<td>The SOAP Action corresponding to the operation to be invoked.</td>
</tr>
<tr>
<td>p_version</td>
<td>The SOAP version, 1.1 or 1.2. The default is 1.1.</td>
</tr>
<tr>
<td>p_collection_name</td>
<td>The name of the collection to store the response.</td>
</tr>
<tr>
<td>p_envelope</td>
<td>The SOAP envelope to post to the service.</td>
</tr>
<tr>
<td>p_username</td>
<td>The username if basic authentication is required for this service.</td>
</tr>
<tr>
<td>p_password</td>
<td>The password if basic authentication is required for this service</td>
</tr>
<tr>
<td>p_scheme</td>
<td>The authentication scheme, Basic (default) or AWS or Digest if supported by your database release.</td>
</tr>
<tr>
<td>p_proxy_override</td>
<td>The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.</td>
</tr>
<tr>
<td>p_transfer_timeout</td>
<td>The amount of time in seconds to wait for a response.</td>
</tr>
<tr>
<td>p_wallet_path</td>
<td>The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.</td>
</tr>
<tr>
<td>p_wallet_pwd</td>
<td>The password to access the wallet.</td>
</tr>
<tr>
<td>p_https_host</td>
<td>The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.</td>
</tr>
</tbody>
</table>

Example

The following example uses the make_request procedure to retrieve a list of movies from a SOAP style Web service. The response is stored in an Application Express collection named MOVIE_LISTINGS.

```sql
declare
    l_envelope CLOB;
BEGIN
  l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<soap:Body>
  <tns:GetTheatersAndMovies>
    <tns:zipCode>43221</tns:zipCode>
    <tns:radius>5</tns:radius>
  </tns:GetTheatersAndMovies>
</soap:Body>
</soap:Envelope>';

apex_web_service.make_request(
  p_url => 'http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx',
  p_action => 'http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
  p_collection_name => 'MOVIE_LISTINGS',
  p_envelope => l_envelope
);```
37.9 MAKE_REQUEST Function

Use this function to invoke a SOAP style Web service with the supplied SOAP envelope returning the results in an XMLTYPE.

Syntax

APEX_WEB_SERVICE.MAKE_REQUEST (  
  p_url IN VARCHAR2,  
  p_action IN VARCHAR2 default null,  
  p_version IN VARCHAR2 default '1.1',  
  p_envelope IN CLOB,  
  p_username IN VARCHAR2 default null,  
  p_password IN VARCHAR2 default null,  
  p_scheme IN VARCHAR2 default 'Basic',  
  p_proxy_override IN VARCHAR2 default null,  
  p_transfer_timeout IN NUMBER default 180,  
  p_wallet_path IN VARCHAR2 default null,  
  p_wallet_pwd IN VARCHAR2 default null,  
  p_https_host IN VARCHAR2 default null,  
  p_credential_static_id IN VARCHAR2 default null,  
  p_token_url IN VARCHAR2 default null )  
RETURN XMLTYPE;

Parameters

Table 37-4  MAKE_REQUEST Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The URL endpoint of the Web service.</td>
</tr>
<tr>
<td>p_action</td>
<td>The SOAP Action corresponding to the operation to be invoked.</td>
</tr>
<tr>
<td>p_version</td>
<td>The SOAP version, 1.1 or 1.2. The default is 1.1.</td>
</tr>
<tr>
<td>p_envelope</td>
<td>The SOAP envelope to post to the service.</td>
</tr>
<tr>
<td>p_username</td>
<td>The username if basic authentication is required for this service.</td>
</tr>
<tr>
<td>p_password</td>
<td>The password if basic authentication is required for this service.</td>
</tr>
<tr>
<td>p_scheme</td>
<td>The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CRED if supported by your database release.</td>
</tr>
<tr>
<td>p_proxy_override</td>
<td>The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.</td>
</tr>
<tr>
<td>p_transfer_timeout</td>
<td>The amount of time in seconds to wait for a response.</td>
</tr>
<tr>
<td>p_wallet_path</td>
<td>The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.</td>
</tr>
<tr>
<td>p_wallet_pwd</td>
<td>The password to access the wallet.</td>
</tr>
</tbody>
</table>
### Table 37-4  (Cont.) MAKE_REQUEST Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_https_host</td>
<td>The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.</td>
</tr>
<tr>
<td>p_credential_static_id</td>
<td>The name of a Web Credential (configured in Shared Components) to be used.</td>
</tr>
<tr>
<td>p_token_url</td>
<td>For token-based authentication flows (like OAuth2): The URL where to get the token from.</td>
</tr>
</tbody>
</table>

### Example

The following example uses the `make_request` function to invoke a SOAP style Web service that returns movie listings. The result is stored in an XMLTYPE.

```sql
declare
  l_envelope CLOB;
  l_xml    XMLTYPE;
BEGIN
  l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<soap:Body>
  <tns:GetTheatersAndMovies>
    <tns:zipCode>43221</tns:zipCode>
    <tns:radius>5</tns:radius>
  </tns:GetTheatersAndMovies>
</soap:Body>
</soap:Envelope>';

  l_xml := apex_web_service.make_request(
    p_url => 'http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx',
    p_action => 'http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
    p_envelope => l_envelope
  );
END
```

### 37.10 MAKE_REST_REQUEST Function

Use this function to invoke a RESTful style Web service supplying either name value pairs, a character based payload or a binary payload and returning the response in a CLOB.

**Syntax**

```sql
APEX_WEB_SERVICE.MAKE_REST_REQUEST(
  p_url                  IN VARCHAR2,
  p_http_method          IN VARCHAR2,
  p_username             IN VARCHAR2 default null,
```
p_password     IN VARCHAR2 default null,
p_scheme       IN VARCHAR2 default 'Basic',
p_proxy_override IN VARCHAR2 default null,
p_transfer_timeout IN NUMBER   default 180,
p_body         IN CLOB default empty_clob(),
p_body_blob    IN BLOB default empty_blob(),
p_parm_name    IN apex_application_global.VC_ARR2 default empty_vc_arr,
p_parm_value   IN apex_application_global.VC_ARR2 default empty_vc_arr,
p_wallet_path IN VARCHAR2 default null,
p_wallet_pwd IN VARCHAR2 default null,
p_https_host IN VARCHAR2 default null,
p_credential_static_id IN VARCHAR2 default null,
p_token_url IN VARCHAR2 default null )
RETURN CLOB;

Parameters

Table 37-5  MAKE_REST_REQUEST Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The URL endpoint of the Web service.</td>
</tr>
<tr>
<td>p_http_method</td>
<td>The HTTP method to use, PUT, POST, GET, HEAD, or DELETE.</td>
</tr>
<tr>
<td>p_username</td>
<td>The username if basic authentication is required for this service.</td>
</tr>
<tr>
<td>p_password</td>
<td>The password if basic authentication is required for this service</td>
</tr>
<tr>
<td>p_scheme</td>
<td>The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CRED if supported by your database release.</td>
</tr>
<tr>
<td>p_proxy_override</td>
<td>The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.</td>
</tr>
<tr>
<td>p_transfer_timeout</td>
<td>The amount of time in seconds to wait for a response.</td>
</tr>
<tr>
<td>p_body</td>
<td>The HTTP payload to be sent as CLOB.</td>
</tr>
<tr>
<td>p_body_blob</td>
<td>The HTTP payload to be sent as binary BLOB. For example, posting a file.</td>
</tr>
<tr>
<td>p_parm_name</td>
<td>The name of the parameters to be used in name/value pairs.</td>
</tr>
<tr>
<td>p_parm_value</td>
<td>The value of the parameters to be used in name/value pairs.</td>
</tr>
<tr>
<td>p_wallet_path</td>
<td>The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.</td>
</tr>
<tr>
<td>p_wallet_pwd</td>
<td>The password to access the wallet.</td>
</tr>
<tr>
<td>p_https_host</td>
<td>The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.</td>
</tr>
<tr>
<td>p_credential_static_id</td>
<td>The name of a Web Credential (configured in Shared Components) to be used.</td>
</tr>
<tr>
<td>p_token_url</td>
<td>For token-based authentication flows (like OAuth2): The URL where to get the token from.</td>
</tr>
</tbody>
</table>
Example

The following example calls a RESTful style Web service using the `make_rest_request` function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared CLOB.

```sql
declare
    l_clob       CLOB;
BEGIN
    l_clob := apex_web_service.make_rest_request(
        p_http_method => 'GET',
        p_parm_name => apex_util.string_to_table('appid:format'),
        p_parm_value => apex_util.string_to_table('xyz:xml'));
END;
```

37.11 MAKE_REST_REQUEST_B Function

Use this function to invoke a RESTful style Web service supplying either name value pairs, a character based payload or a binary payload and returning the response in a BLOB.

Syntax

```sql
APEX_WEB_SERVICE.MAKE_REST_REQUEST_B(  
    p_url                  IN VARCHAR2,
    p_http_method          IN VARCHAR2,
    p_username             IN VARCHAR2 default null,
    p_password             IN VARCHAR2 default null,
    p_scheme               IN VARCHAR2 default 'Basic',
    p_proxy_override       IN VARCHAR2 default null,
    p_transfer_timeout     IN NUMBER   default 180,
    p_body                 IN CLOB     default empty_clob(),
    p_body_blob            IN BLOB     default empty_blob(),
    p_parm_name            IN apex_application_global.VC_ARR2 default empty_vc_arr,
    p_parm_value           IN apex_application_global.VC_ARR2 default empty_vc_arr,
    p_wallet_path          IN VARCHAR2 default null,
    p_wallet_pwd           IN VARCHAR2 default null,
    pHttps_host            IN VARCHAR2 default null,
    p_credential_static_id IN VARCHAR2 default null,
    p_token_url            IN VARCHAR2 default null )
RETURN BLOB;
```
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_url</td>
<td>The URL endpoint of the Web service.</td>
</tr>
<tr>
<td>p_http_method</td>
<td>The HTTP method to use, PUT, POST, GET, HEAD, or DELETE.</td>
</tr>
<tr>
<td>p_username</td>
<td>The username if basic authentication is required for this service.</td>
</tr>
<tr>
<td>p_password</td>
<td>The password if basic authentication is required for this service.</td>
</tr>
<tr>
<td>p_scheme</td>
<td>The authentication scheme, Basic (default) or AWS or Digest or</td>
</tr>
<tr>
<td></td>
<td>OAUTH_CLIENT_CRED if supported by your database release.</td>
</tr>
<tr>
<td>p_proxy_override</td>
<td>The proxy to use for the request. The proxy supplied overrides the proxy</td>
</tr>
<tr>
<td></td>
<td>defined in the application attributes.</td>
</tr>
<tr>
<td>p_transfer_timeout</td>
<td>The amount of time in seconds to wait for a response.</td>
</tr>
<tr>
<td>p_body</td>
<td>The HTTP payload to be sent as CLOB.</td>
</tr>
<tr>
<td>p_body_blob</td>
<td>The HTTP payload to be sent as binary BLOB. For example, posting a file.</td>
</tr>
<tr>
<td>p_parm_name</td>
<td>The name of the parameters to be used in name/value pairs.</td>
</tr>
<tr>
<td>p_parm_value</td>
<td>The value of the parameters to be used in name/value pairs.</td>
</tr>
<tr>
<td>p_wallet_path</td>
<td>The file system path to a wallet if the URL endpoint is https. For example,</td>
</tr>
<tr>
<td></td>
<td>file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet</td>
</tr>
<tr>
<td></td>
<td>defined in the instance settings.</td>
</tr>
<tr>
<td>p_wallet_pwd</td>
<td>The password to access the wallet.</td>
</tr>
<tr>
<td>p_https_host</td>
<td>The host name to be matched against the common name (CN) of the remote</td>
</tr>
<tr>
<td></td>
<td>server’s certificate for an HTTPS request.</td>
</tr>
<tr>
<td>p_credential_static_id</td>
<td>The name of a Web Credential (configured in Shared</td>
</tr>
<tr>
<td></td>
<td>Components) to be used.</td>
</tr>
<tr>
<td>p_token_url</td>
<td>For token-based authentication flows (like OAuth2): The URL</td>
</tr>
<tr>
<td></td>
<td>where to get the token from.</td>
</tr>
</tbody>
</table>

### Example

The following example calls a RESTful style Web service using the make_rest_request function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared BLOB.

```sql
declare
  l_blob     BLOB;
BEGIN
  l_blob := apex_web_service.make_rest_request_b(  
      p_http_method => 'GET',
      p_parm_name => apex_util.string_to_table('appid:format'),
      p_parm_value => apex_util.string_to_table('xyz:xml'));
```
37.12 OAUTH_AUTHENTICATE Function

This function performs OAUTH authentication and requests an OAuth access token. The token and its expiry date are stored in the global variable `g_oauth_token`.

```plaintext
type oauth_token is record(
    token      varchar2(255),
    expires    date );
```

**Note:**
Currently only the Client Credentials flow is supported.

**Syntax**

```plaintext
APEX_WEB_SERVICE.OAUTH_AUTHENTICATE(
    p_token_url         in varchar2,
    p_client_id         in varchar2,
    p_client_secret     in varchar2,
    p_flow_type         in varchar2 default OAUTH_CLIENT_CRED,
    p_proxy_override    in varchar2 default null,
    p_transfer_timeout  in number   default 180,
    p_wallet_path       in varchar2 default null,
    p_wallet_pwd        in varchar2 default null,
    p_https_host        in varchar2 default null );
```

**Parameters**

**Table 37-7  OAUTH_AUTHENTICATE Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_token_url</td>
<td>The url endpoint of the OAuth token service.</td>
</tr>
<tr>
<td>p_client_id</td>
<td>OAuth Client ID to use for authentication.</td>
</tr>
<tr>
<td>p_client_secret</td>
<td>OAuth Client Secret to use for authentication.</td>
</tr>
<tr>
<td>p_flow_type</td>
<td>OAuth flow type - only OAUTH_CLIENT_CRED is supported at this time.</td>
</tr>
<tr>
<td>p_proxy_override</td>
<td>The proxy to use for the request.</td>
</tr>
<tr>
<td>p_transfer_timeout</td>
<td>The amount of time in seconds to wait for a response.</td>
</tr>
<tr>
<td>p_wallet_path</td>
<td>The filesystem path to a wallet if request is https. For example, file:/usr/home/oracle/WALLETS.</td>
</tr>
<tr>
<td>p_wallet_pwd</td>
<td>The password to access the wallet.</td>
</tr>
</tbody>
</table>
Table 37-7 (Cont.) OAUTH_AUTHENTICATE Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_https_host</td>
<td>The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.</td>
</tr>
</tbody>
</table>

Example

```
begin
  apex_web_service.oauth_authenticate(
    p_token_url => 'http://{host}:{port}/ords/.../oauth/token',
    p_client_id => 'client-id',
    p_client_secret => 'client-secret'
  );
end;
```

37.13 OAUTH_GET_LAST_TOKEN Function

This function returns the OAuth access token received in the last OAUTH_AUTHENTICATE call. Returns NULL when the token is already expired or OAUTH_AUTHENTICATE has not been called.

Returns

The OAuth access token from the last OAUTH_AUTHENTICATE call; NULL when the token is expired.

Syntax

```
function oauth_get_last_token return varchar2;
```

Example

```
select apex_web_service.oauth_get_last_token from dual;
```

37.14 OAUTH_SET_TOKEN Function

This function sets the OAuth Access token to be used in subsequent MAKE_REST_REQUEST calls. This procedure can be used to set a token which has been obtained by other means than with OAUTH_AUTHENTICATE (for instance, custom code).

Parameters

Table 37-8 OAUTH_SET_TOKEN Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_token</td>
<td>The OAuth access token to be used by MAKE_REST_REQUEST calls.</td>
</tr>
</tbody>
</table>
### Table 37-8  (Cont.) OAUTH_SET_TOKEN Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_expires</td>
<td>Optional: The token expiry date; NULL means: No expiration date.</td>
</tr>
</tbody>
</table>

#### Example

```sql
begin
  apex_web_service.oauth_set_token(
    p_token => '{oauth access token}'
  );
end;
```

### 37.15 PARSE_RESPONSE Function

Use this function to parse the response from a Web service that is stored in a collection and return the result as a VARCHAR2 type.

#### Syntax

```sql
APEX_WEB_SERVICE.PARSE_RESPONSE(
  p_collection_name IN VARCHAR2,
  p_xpath             IN VARCHAR2,
  p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection where the Web service response is stored.</td>
</tr>
<tr>
<td>p_xpath</td>
<td>The XPath expression to the desired node.</td>
</tr>
<tr>
<td>p_ns</td>
<td>The namespace to the desired node.</td>
</tr>
</tbody>
</table>

#### Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared VARCHAR2 variable.

```sql
declare
  l_response_msg  VARCHAR2(4000);
BEGIN
  l_response_msg := apex_web_service.parse_response(
    p_collection_name=>'STELLENT_CHECKIN',
    p_xpath =>
      '//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/'
  );
end;
```
37.16 PARSE_RESPONSE_CLOB Function

Use this function to parse the response from a Web service that is stored in a collection and return the result as a CLOB type.

Syntax

```sql
APEX_WEB_SERVICE.PARSE_RESPONSE_CLOB (  
    p_collection_name   IN VARCHAR2,  
    p_xpath             IN VARCHAR2,  
    p_ns                IN VARCHAR2 default null )  
RETURN CLOB;
```

Parameters

**Table 37-10  PARSE_RESPONSE_CLOB Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_collection_name</td>
<td>The name of the collection where the Web service response is stored.</td>
</tr>
<tr>
<td>p_xpath</td>
<td>The XPath expression to the desired node.</td>
</tr>
<tr>
<td>p_ns</td>
<td>The namespace to the desired node.</td>
</tr>
</tbody>
</table>

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared CLOB variable.

```sql
declare
    l_response_msg  CLOB;
BEGIN
    l_response_msg := apex_web_service.parse_response_clob(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath='//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/idc:statusMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');
END;
```

37.17 PARSE_XML Function

Use this function to parse the response from a Web service returned as an XMLTYPE and return the value requested as a VARCHAR2.
### Syntax

APEX_WEB_SERVICE.PARSE_XML (  
    p_xml               IN XMLTYPE,  
    p_xpath             IN VARCHAR2,  
    p_ns                IN VARCHAR2 default null )  
RETURN VARCHAR2;

### Parameters

#### Table 37-11  PARSE_XML Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_xml</td>
<td>The XML document as an XMLTYPE to parse.</td>
</tr>
<tr>
<td>p_xpath</td>
<td>The XPath expression to the desired node.</td>
</tr>
<tr>
<td>p_ns</td>
<td>The namespace to the desired node.</td>
</tr>
</tbody>
</table>

### Example

The following example uses the `make_request` function to call a Web service and store the results in a local XMLTYPE variable. The `parse_xml` function is then used to pull out a specific node of the XML document stored in the XMLTYPE and stores it in a locally declared VARCHAR2 variable.

```sql
declare  
  l_envelope CLOB;  
  l_xml XMLTYPE;  
  l_movie VARCHAR2(4000);  
BEGIN  
  l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmllns:tns="http://www.ignyte.com/whatsshowing"
xmllns:xs="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <tns:GetTheatersAndMovies>
      <tns:zipCode>43221</tns:zipCode>
      <tns:radius>5</tns:radius>
    </tns:GetTheatersAndMovies>
  </soap:Body>
</soap:Envelope>';  

  l_xml := apex_web_service.make_request(  
    p_url => 'http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx',  
    p_action => 'http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',  
    p_envelope => l_envelope );  

  l_movie := apex_web_service.parse_xml(  
    p_xml => l_xml,  
    p_xpath => '/GetTheatersAndMoviesResponse'  
  );
```
37.18 PARSE_XML_CLOB Function

Use this function to parse the response from a Web service returned as an XMLTYPE and return the value requested as a CLOB.

Syntax

APEX_WEB_SERVICE.PARSE_XML_CLOB (  
  p_xml               IN XMLTYPE,  
  p_xpath             IN VARCHAR2,  
  p_ns                IN VARCHAR2 default null )  
RETURN VARCHAR2;

Parameters

Table 37-12  PARSE_XML_CLOB Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_xml</td>
<td>The XML document as an XMLTYPE to parse.</td>
</tr>
<tr>
<td>p_xpath</td>
<td>The XPath expression to the desired node.</td>
</tr>
<tr>
<td>p_ns</td>
<td>The namespace to the desired node.</td>
</tr>
</tbody>
</table>

Example

The following example uses the make_request function to call a Web service and store the results in a local XMLTYPE variable. The parse_xml function is then used to pull out a specific node of the XML document stored in the XMLTYPE and stores it in a locally declared VARCHAR2 variable.

declare  
  l_envelope CLOB;  
  l_xml XMLTYPE;  
  l_movie CLOB;
BEGIN  
  l_envelope := '<xml version="1.0" encoding="UTF-8"?>  
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
  xmlns:tns="http://www.ignyte.com/whatsshowing"  
  xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <soap:Body>  
  <tns:GetTheatersAndMovies>  
  <tns:zipCode>43221</tns:zipCode>  
  <tns:radius>5</tns:radius>  
  </tns:GetTheatersAndMovies>  
  </soap:Body>  
  </soap:Envelope>';
l_xml := apex_web_service.make_request(
    p_url => 'http://www.ignyte.com/webservices/
        ignyte.whatsshowing.webservice/moviefunctions.asmx',
    p_action => 'http://www.ignyte.com/whatsshowing/
        GetTheatersAndMovies',
    p_envelope => l_envelope );

l_movie := apex_web_service.parse_xml_clob(
    p_xml => l_xml,
    p_xpath => ' //GetTheatersAndMoviesResponse/
        GetTheatersAndMoviesResult/Theater/Movies/Movie/Name[1]',
    p_ns => ' xmlns="http://www.ignyte.com/whatsshowing"' );

END;
APEX_ZIP

This package allows to compress and to uncompress files and store them in a ZIP file.

- Data Types
- ADD_FILE Procedure
- FINISH Procedure
- GET_FILE_CONTENT Function
- GET_FILES Function

38.1 Data Types

The data types used by the APEX_ZIP package are described in this section.

t_files

```plaintext
type t_files is table of varchar2(32767) index by binary_integer;
```

38.2 ADD_FILE Procedure

This procedure adds a single file to a zip file.

```plaintext
APEX_ZIP.ADD_FILE (  
    p_zipped_blob IN OUT NOCOPY BLOB,  
    p_file_name   IN VARCHAR2,  
    p_content     IN BLOB );
```

Parameters

Table 38-1 ADD_FILE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_zipped_blob</td>
<td>BLOB containing the zip file.</td>
</tr>
<tr>
<td>p_file_name</td>
<td>File name, including path, of the file to be added to the zip file.</td>
</tr>
</tbody>
</table>
### Table 38-1 (Cont.) ADD_FILE Procedure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_content</td>
<td>BLOB containing the file.</td>
</tr>
</tbody>
</table>

### Example

This example reads multiple files from a table and puts them into a single zip file.

```sql
declare
    l_zip_file blob;
begin
    for l_file in ( select file_name, file_content from my_files )
    loop
        apex_zip.add_file ( p_zipped_blob => l_zip_file, p_file_name   => l_file.file_name, p_content => l_file.file_content );
    end loop;
    apex_zip.finish ( p_zipped_blob => l_zip_file );
end;
```

### 38.3 FINISH Procedure

This procedure completes the creation of a zip file after adding files with APEX_ZIP.ADD_FILE.

#### Syntax

APEX_ZIP.FINISH ( p_zipped_blob IN OUT NOCOPY BLOB );

#### Parameters

*Table 38-2 FINISH Procedure Parameters*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_zipped_blob</td>
<td>BLOB containing the zip file.</td>
</tr>
</tbody>
</table>

#### Example

See "ADD_FILE Procedure" for an example.
38.4 GET_FILE_CONTENT Function

This function returns the BLOB of a file contained in a provided zip file.

**Syntax**

```sql
APEX_ZIP.GET_FILE_CONTENT (  
    p_zipped_blob IN BLOB,
    p_file_name IN VARCHAR2,
    p_encoding IN VARCHAR2 DEFAULT NULL )
RETURN BLOB;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_zipped_blob</td>
<td>This is the BLOB containing the zip file.</td>
</tr>
<tr>
<td>p_file_name</td>
<td>File name, including path, of a file located in the zip file.</td>
</tr>
<tr>
<td>p_encoding</td>
<td>Encoding used to zip the file.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>BLOB containing the zip file.</td>
</tr>
</tbody>
</table>

**Example**

See "GET_FILES Function" for an example.

38.5 GET_FILES Function

This function returns an array of file names, including the path, of a provided zip file that contains a BLOB.

**Syntax**

```sql
APEX_ZIP.GET_FILES (  
    p_zipped_blob IN BLOB,
    p_only_files IN BOOLEAN DEFAULT TRUE,
    p_encoding IN VARCHAR2 DEFAULT NULL )
RETURN t_files;
```
Parameters

### Table 38-5  GET_FILES Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_zipped_blob</td>
<td>This is the zip file containing the BLOB.</td>
</tr>
<tr>
<td>p_only_files</td>
<td>If set to TRUE, empty directory entries are not included in the returned array. Otherwise, set to FALSE to include empty directory entries.</td>
</tr>
<tr>
<td>p_encoding</td>
<td>This is the encoding used to zip the file.</td>
</tr>
</tbody>
</table>

Returns

### Table 38-6  GET_FILES Function Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_files</td>
<td>A table of file names and path. See &quot;Data Types&quot; for more details.</td>
</tr>
</tbody>
</table>

Example

This example demonstrates reading a zip file from a table, extracting it and storing all files of the zip file into my_files.

```sql
declare
    l_zip_file      blob;
    l_unzipped_file blob;
    l_files         apex_zip.t_files;
begin
    select file_content into l_zip_file from my_zip_files where file_name = 'my_file.zip';
    l_files := apex_zip.get_files ( p_zipped_blob => l_zip_file );
    for i in 1 .. l_files.count loop
        l_unzipped_file := apex_zip.get_file_content ( p_zipped_blob => l_zip_file,
                                                      p_file_name   => l_files(i) );
        insert into my_files ( file_name, file_content ) values ( l_files(i), l_unzipped_file );
    end loop;
end;
```
This content of this chapter has been moved to the Oracle Application Express JavaScript API Reference.
Using REST Administration Interface API

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

Tip:
The REST Administration Interface requires Oracle REST Data Services (ORDS) release 3.0.5 or later.

- Authentication
- Individual REST Services

See Also:
"Using the REST Administration Interface to View Usage Statistics" in Oracle Application Express Administration Guide

40.1 Authentication

REST clients must authenticate before accessing the administrative REST services. First, an Oracle Application Express instance administrator must log into the Oracle Application Express application and register a REST client.

When a client has been registered in Instance Administration, the dialog shows Client ID and Client Secret, with which the client can then perform authentication following the OAuth2 Client Credentials flow. A client first connects with a Client ID and a Client Secret as the credentials. Upon successful authentication, the server sends back the OAuth Access Token. Using this access token, the client can then access the administrative REST services.
HTTP Request Syntax Parameter

Table 40-1 HTTP Request Syntax

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method</td>
<td>POST</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://application-express-host:port/ords/apex_instance_admin_user/oauth/token">http://application-express-host:port/ords/apex_instance_admin_user/oauth/token</a></td>
</tr>
<tr>
<td>Request Body</td>
<td>grant_type=client_credentials</td>
</tr>
<tr>
<td>HTTP Request Headers</td>
<td>&quot;Content-Type&quot;: &quot;application/x-www-form-urlencoded&quot; &quot;Authorization&quot;: Client-ID:Client Secret in Base64-encoded form</td>
</tr>
</tbody>
</table>

Returns

Returns a JSON object with the following structure upon successful authentication:

```json
{
  "access_token": OAuth access token for subsequent requests,
  "token_type": "bearer",
  "expires_in": lifetime of the OAuth token, in seconds; typically "3600"
}
```

If authentication is unsuccessful, the server responds with HTTP-401:Unauthorized.

Examples

In the following example ClientID stands for the Client ID and ClientSecret for the Client Secret.

Example 1

The example displays the following output when you execute command line utility curl:

```
$ curl -i
--user ClientId:ClientSecret
--data "grant_type=client_credentials"
http://application-express-host:port/ords/apex_instance_admin_user/oauth/token

HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

"access_token":"LfXJilIBdzj5JPRn4xb5QQ..","token_type":"bearer","expires_in":3600
```
Use a JSON parser to extract the value of the `access_token` attribute and use it in subsequent requests.

**Example 2**

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

```sql
begin
    apex_web_service.oauth_authenticate(
        p_token_url => 'http://application-express-host:port/ords/apex_instance_admin_user/oauth/token',
        p_client_id => 'ClientId',
        p_client_secret => 'ClientSecret'
    );
    dbms_output.put_line( 'The token is: ' || apex_web_service.oauth_get_last_token );
end;
/
```

The token is: LfXJillIBdzj5JPRn4xb5QQ..

With the acquired OAuth Access Token, the administrative REST Services can be called.

**See Also:**

*Oracle Application Express Administration Guide*

### 40.2 Individual REST Services

Individual REST Services have the following services:

- Fetch Instance-Level Statistics
- Fetch Workspace-Level Statistics
- Application-Level Statistics
- Instance Overview
- REST Service Version Information

#### 40.2.1 Fetch Instance-Level Statistics

This service returns usage statistics for the whole Oracle Application Express instance.
HTTP Request Syntax Parameter

Table 40-2   HTTP Request Syntax

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/instance">http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/instance</a></td>
</tr>
<tr>
<td>HTTP Request Headers</td>
<td>&quot;authorization&quot;: &quot;Bearer: OAuth access token acquired with Authentication&quot;</td>
</tr>
</tbody>
</table>

Returns

```json
{
  "items": [
   {
     "log_day": "2016-09-15T00:00:00Z",
     "workspace_id": 1809074264671554,
     "workspace_name": "SOMEWORKSPACE",
     "workspace_link": {
       "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/someworkspace"
     },
     "application_id": 4750,
     "application_name": "Oracle APEX Packaged Applications",
     "application_link": {
   
```

Parameters

Filtering is possible for each of the JSON attributes by appending a query to the request URL as follows:

```
http://application-express-host.../latest/instance?q=query
```

Examples

Get All Instance Statistics

The example displays the following output when you execute command line utility `curl`:

```
http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/instance
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:
The example assumes that the OAUTH_AUTHENTICATE procedure is successful.

```sql
select apex_web_service.make_rest_request(
    p_url => 'http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/instance'
    p_http_method => 'GET' )
from dual;
```

The following is a JSON-Document response:

```json
{
    "items": [
        {
            "log_day": "2016-09-15T00:00:00Z",
            "workspace_id": 267781782378434879,
            "workspace_name": "SOMEWORKSPACE",
            "workspace_link": {
                "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/someworkspace"
            }
        }
    ]
}
```

Get Instance Statistics Since August 1st, 2016

The example displays the following output when you execute command line utility `curl`:

```
$ curl -H"Authorization: Bearer LfXJiIlIbdzj5JPRn4xb5QQ..
   -i http://localhost:8081/ords/restauth/emp/list/
http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/instance?q=%7B%22log_day%22:%222016-08-01T00:00:00Z%22%22%c2%a0%7B%22q%22:%22%7B%22%c2%a0%7B%22%7D%7D
```

The example displays the following output when you use the APEX_WEB_SERVICE package in another Application Express instance:

```sql
select apex_web_service.make_rest_request(
    p_url => 'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/instance?q=||
    utl_url.escape('{"log_day": "$gt": {"$date": "2016-08-01T00:00:00Z"}}')
),
```

(The example assumes that the OAUTH_AUTHENTICATE procedure is successful.)
The following is a JSON-Document response:

```
{
   "items": [
      {
         "log_day": "2016-09-15T00:00:00Z",
         "workspace_id": 267781782378434879,
         "workspace_name": "SOMEWORKSPACE",
         "workspace_link": {
            "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/someworkspace"
         }
      }
   ]
}
```

### See Also:

*Oracle REST Data Services Installation, Configuration, and Development Guide* for more information on how to build a query.

## 40.2.2 Fetch Workspace-Level Statistics

This service returns usage statistics for a specific Application Express workspace.

### HTTP Request Syntax Parameter

**Table 40-3 HTTP Request Syntax**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTTP Method</strong></td>
<td>GET</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><code>http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/workspace-name</code></td>
</tr>
<tr>
<td><strong>HTTP Request Headers</strong></td>
<td>&quot;authorization&quot;: &quot;Bearer: OAuth access token acquired with Authentication&quot;</td>
</tr>
</tbody>
</table>

### Returns

```
{
   "items": [
      {
         "log_day": "2016-09-15T00:00:00Z",
         "workspace_id": 1809919633676005,
         "workspace_name": "SOMEWORKSPACE",
         "workspace_link": {
            "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/someworkspace"
         }
      }
   ]
}
```
Parameters

Filtering is possible for each of the JSON attributes by appending a query to the request URL as follows:

```
http://application-express-host.../latest/instance?q=query
```

Examples

Get All Statistics For Workspace "MYWORKSPACE"

The example displays the following output when you execute command line utility curl:

```
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

```
Note:
The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.
```

```
select apex_web_service.make_rest_request(
    p_url => 'http://application-express-host:port/ords/
apex_instance_admin_user/stats/latest/workspace/myworkspace',
    p_http_method => 'GET',
    p_scheme => 'OAUTH_CLIENT_CRED' )
from dual;
```

The following is a JSON-Document response:

```
{
    "items": [
        {
            "log_day": "2016-09-15T00:00:00Z",
            "workspace_id": 1809919633676005,
            "application_id": 106,
        }
    ]
}
```
Get Workspace Statistics For "MYWORKSPACE" Since August 1st, 2016

The example displays the following output when you execute command line utility curl:

```bash
$ curl -H"Authorization: Bearer LfXJ11IBdzj5JPRn4xb5QQ.." 
http://localhost:8081/ords/restauth/emp/list/ 
http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/myworkspace?q=%7B%22log_day%22:%7B%22$gt%22:%7B%22$date%22:%222016-08-01T00:00:00Z%22%7D%7D
```

The example displays the following output when you use the APEX_WEB_SERVICE package in another Application Express instance:

```sql
Note:
The example assumes that the OAUTH_AUTHENTICATE procedure is successful.

```sql
select apex_web_service.make_rest_request(
    p_url => http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/workspace/myworkspace?q=\$%22log_day%22:%222016-08-01T00:00:00Z\$
    utl_url.escape("log_day":"$gt":"$date":"2016-08-01T00:00:00Z" ),
    p_http_method => 'GET' )
from dual;
```

The following is a JSON-Document response:

```json
{
    "items": [
    
    
    "log_day": "2016-09-15T00:00:00Z",
    "workspace_id": 1809919633676005,
    "application_id": 106,
    "application_name": "Sample Calendar",
    "application_link": { 
        "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/106"
    },
    "page_events": 94,
}
```
40.2.3 Application-Level Statistics

This service returns usage statistics for a specific application.

HTTP Request Syntax Parameter

Table 40-4 HTTP Request Syntax

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/application-id">http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/application-id</a></td>
</tr>
<tr>
<td>HTTP Request Headers</td>
<td>&quot;authorization&quot;: &quot;Bearer: OAuth access token acquired with Authentication&quot;</td>
</tr>
</tbody>
</table>

Returns

```
{
  "items": [
    {
      "log_day": "2016-09-15T00:00:00Z",
      "workspace_id": 1809919633676005,
      "application_id": 106,
      "application_name": "Sample Calendar",
      "application_link": {
        "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/106"
      },
      "page_events": 94,
    }
  ]
```

Parameters

Filtering is possible for each of the JSON attributes by appending a query to the request URL as follows:

```
http://application-express-host.../stats/latest/application/application-id?q=query
```
Examples

Get All Statistics For Application "106"

The example displays the following output when you execute command line utility curl:

```bash
```

The example displays the following output when you use the APEX_WEB_SERVICE package in another Application Express instance:

```
Note:
The example assumes that the OAUTH_AUTHENTICATE procedure is successful.
```

```sql
select apex_web_service.make_rest_request(  
    p_url => 'http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/106',  
    p_http_method => 'GET',  
    p_scheme => 'OAUTH_CLIENT_CRED' )  
from dual;
```

The following is a JSON-Document response:

```json
{
    "items": [
        {
            "log_day": "2016-09-15T00:00:00Z",
            "workspace_id": 180991963676005,
            "application_id": 106,
            "application_name": "Sample Calendar",
            "application_link": {
                "$ref": "http://application-express-host:port/ords/apex_instance_admin_user/stats/latest/application/106"
            },
            "page_events": 94,

Get Statistics For Application "106" Since August 1st, 2016

The example displays the following output when you execute command line utility curl:

```bash
```
The example displays the following output when you use the APEX_WEB_SERVICE package in another Application Express instance:

```sql
select apex_web_service.make_rest_request(
    p_url =>         'http://application-express-host:port/ords/
    apex_instance_admin_user/stats/latest/application/106?q=© ||
    utl_url.escape( "log_day": "$gt": "$date":
    "2016-08-01T00:00:00Z" ),
    p_http_method => 'GET' )
from dual;
```

The following is a JSON-Document response:

```
{
   "items": [ 
    {
      "log_day": "2016-09-15T00:00:00Z",
      "workspace_id": 1809919633676005,
      "application_id": 106,
      "application_name": "Sample Calendar",
      "application_link": {
        "$ref": "http://application-express-host:port/ords/
    apex_instance_admin_user/stats/latest/application/106"
      },
      "page_events": 94,
    }
   ]
}
```

See Also:

Oracle REST Data Services Installation, Configuration, and Development Guide for more information on how to build a query.

40.2.4 Instance Overview

This service returns aggregated overview data for an Application Express instance.
HTTP Request Syntax Parameter

Table 40-5  HTTP Request Syntax

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
<tr>
<td>URL</td>
<td><code>http://application-express-host:port/ords/apex_instance_admin_user/info/latest/instance/number-of-days</code></td>
</tr>
<tr>
<td>HTTP Request Headers</td>
<td>&quot;authorization&quot;: &quot;Bearer: OAuth access token acquired with Authentication&quot;</td>
</tr>
</tbody>
</table>

Returns

```
{
  "items": [
    {
      "workspaces_total": 1074,
      "apps_total": 2827,
      "schemas_total": 1065,
      "reporting_timeframe_since": "2016-07-16T13:30:40Z",
      "reporting_timeframe_to": "2016-10-14T13:30:40Z",
      "active_apps_timeframe": 2827,
      "active_developers_timeframe": 731,
      "workspaces_timeframe": 1074
    }
  ],
  "first": {
    "$ref": "https://apexea.oracle.com/pls/apex/apex_instance_admin_user/info/latest/instance/90"
  }
}
```

Parameters

`number-of-days`: Return the aggregated values from today to the given number of days into the past.

Example

The example displays the following output when you execute command line utility `curl`:

```
http://application-express-host:port/ords/apex_instance_admin_user/info/latest/instance/90
```
The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

```sql
select apex_web_service.make_rest_request(
    p_url => 'http://application-express-host:port/ords/apex_instance_admin_user/info/latest/instance/90',
    p_http_method => 'GET' )
from dual;
```

The following is a JSON-Document response:

```json
{
    "items": [
        {
            "workspaces_total": 1074,
            "apps_total": 2827,
            "schemas_total": 1065,
            "reporting_timeframe_since": "2016-07-16T13:30:40Z",
            "reporting_timeframe_to": "2016-10-14T13:30:40Z",
            "active_apps_timeframe": 2827,
            "active_developers_timeframe": 731,
            "workspaces_timeframe": 1074
        }
    ],
    "first": {
        "$ref": "https://apexea.oracle.com/pls/apex/apex_instance_admin_user/info/latest/instance/90"
    }
}
```

### 40.2.5 REST Service Version Information

This service returns version information for the REST Administrative Interface.

**HTTP Request Syntax Parameter**

**Table 40-6  HTTP Request Syntax**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
</tbody>
</table>
### Table 40-6  (Cont.) HTTP Request Syntax

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td><a href="http://application-express-host:port/ords/apex_instance_admin_user/info/latest/">http://application-express-host:port/ords/apex_instance_admin_user/info/latest/</a></td>
</tr>
<tr>
<td>HTTP Request Headers</td>
<td>&quot;authorization&quot;: &quot;Bearer: OAuth access token aquired with Authentication&quot;</td>
</tr>
</tbody>
</table>

### Returns

```
{
    "items": [
        {
            "version": "5.1.0"
        }
    ]
}
```

### Example

The example displays the following output when you execute command line utility `curl`:

```
$ curl -H"Authorization: Bearer LfXJi1IBdzj5JPRn4yb5QQ..
-i http://localhost:8081/ords/restauth/emp/list/
http://application-express-host:port/ords/apex_instance_admin_user/info/rest-service-version
```

The example displays the following output when you use the `APEX_WEB_SERVICE` package in another Application Express instance:

```sql
select apex_web_service.make_rest_request(
    p_url => 'http://application-express-host:port/ords/apex_instance_admin_user/info/latest/rest-service-version',
    p_http_method => 'GET')
from dual;
```

---

**Note:**

The example assumes that the `OAUTH_AUTHENTICATE` procedure is successful.
The following is a JSON-Document response:

```json
{
   "items": [
      {
         "version": "5.1.0"
      }
   ]
}
```
Index

A

ADD_COLUMN Procedure, 15-8
ADD_DML_ROW Procedure, 15-9
ADD_FILTER Procedure, 15-10
ADD_ORDER_BY Procedure, 15-14
ADD_PARAMETER Procedure, 15-15
APEX_ACL, 2-1
  ADD_USER_ROLE procedure signature 1, 2-1
  ADD_USER_ROLE procedure signature 2, 2-2
HAS_USER_ANY_ROLES function, 2-3
HAS_USER_ROLE function, 2-3
REMOVE_ALL_USER_ROLES procedure, 2-7
REMOVE_USER_ROLE procedure signature 1, 2-4
REMOVE_USER_ROLE procedure signature 2, 2-5
REPLACE_USER_ROLES procedure signature 1, 2-6
REPLACE_USER_ROLES procedure signature 2, 2-7
APEX_APP_SETTING, 6-1
  GET_VALUE Function, 6-2
  SET_VALUE Procedure, 6-1
APEX_APPLICATION
  global variables, 1-1
  HELP Procedure, 1-4
  package, 1-1
  Referencing Arrays, 1-2
  STOP_APEX_ENGINE Procedure, 1-6
APEX_APPLICATION_INSTALL, 3-1
  CLEAR_ALL procedure, 3-5
  example, 3-3
  examples, 3-3
  GENERATE_APPLICATION_ID procedure, 3-5
  GENERATE_OFFSET procedure, 3-6
  GET_APPLICATION_ALIAS function, 3-7
  GET_APPLICATION_ID function, 3-7
  GET_APPLICATION_NAME function, 3-8
  GET_IMAGE_PREFIX function, 3-9
  GET_KEEP_SESSIONS function, 3-10
  APEX_APPLICATION_INSTALL (continued)
    GET_NO_PROXY_DOMAINS function, 3-10
    GET_OFFSET function, 3-11
    GET_PROXY function, 3-12
    GET_REMOTE_SERVER_BASE_URL function, 3-12
    GET_REMOTE_SERVER_HTTPS_HOST function, 3-13
    GET_SCHEMA function, 3-14
    GET_WORKSPACE_ID function, 3-14
    import application into different workspaces, 3-3
    import application with generated app ID, 3-3
    import application with specified app ID, 3-3
    import application without modification, 3-3
    import into training instance, 3-3
    import script examples, 3-3
    package overview, 3-2
    SET_APPLICATION_ALIAS procedure, 3-15
    SET_APPLICATION_ID procedure, 3-16
    SET_APPLICATION_NAME procedure, 3-16
    SET_IMAGE_PREFIX procedure, 3-18
    SET_KEEP_SESSIONS procedure, 3-19
    SET_OFFSET procedure, 3-19
    SET_PROXY procedure, 3-20
    SET_REMOTE_SERVER procedure, 3-21
    SET_SCHEMA procedure, 3-22
    SET_WORKSPACE_ID procedure, 3-23
    SET_WORKSPACE Procedure, 3-23
    APEX_APPLICATION_INSTALL.GET_AUTO_INSTALL_SUP_OBJ, 3-9
    APEX_APPLICATION_INSTALL.SET_AUTO_INSTALL_SUP_OBJ, 3-17
APEX_AUTHENTICATION, 4-1
  CALLBACK 1 Procedure, 4-3
  CALLBACK 2 Procedure, 4-3
  CALLBACK Procedure, 4-1
  GET_CALLBACK_URL Procedure, 4-4
  GET_LOGIN_USERNAME_COOKIE_LOGIN Function, 4-5
  IS_AUTHENTICATED Function, 4-6
  IS_PUBLIC_USER Function, 4-6
  LOGIN Procedure, 4-7
  LOGOUT Procedure, 4-8
  POST_LOGIN Procedure, 4-8
APEX_AUTHENTICATION (continued)
  SEND_LOGIN_USERNAME_COOKIE
    Procedure, 4-9
APEX_AUTHORIZATION, 5-1
  ENABLE_DYNAMIC_GROUPS procedure, 5-1
  IS_AUTHORIZED function, 5-2
  RESET_CACHE procedure, 5-3
APEX_COLLECTION, 7-2
  ADD_MEMBER function, 7-13
  ADD_MEMBER procedure, 7-11
  ADD_MEMBERS procedure, 7-14
  COLLECTION_exists function, 7-16
  COLLECTION_HAS_CHANGED function, 7-17
  COLLECTION_MEMBER_COUNT function, 7-18
  CREATE_COLLECTION procedure, 7-18
  CREATE_COLLECTION_FROM_QUERY, 7-20
  CREATE_COLLECTION_FROM_QUERY_B
    procedure, 7-23
  CREATE_COLLECTION_FROM_QUERY_B
    procedure (No bind version), 7-25
  CREATE_COLLECTION_FROM_QUERY2
    procedure, 7-22
  CREATE_COLLECTION_FROM_QUERYB2
    procedure, 7-26
  CREATE_COLLECTION_FROM_QUERYB2
    procedure (No bind version), 7-28
  CREATE_OR_TRUNCATE_COLLECTION
    procedure, 7-30
  DELETE_ALL_COLLECTIONS procedure, 7-30
  DELETE_ALL_COLLECTIONS_SESSION
    procedure, 7-30
  DELETE_COLLECTION procedure, 7-31
  DELETE_MEMBER procedure, 7-32
  DELETE_MEMBERS procedure, 7-33
  GET_MEMBER_MD5 function, 7-34
  MERGE_MEMBERS procedure, 7-35
  MOVE_MEMBER_DOWN procedure, 7-37
  MOVE_MEMBER_UP procedure, 7-38
  RESEQUENCE_COLLECTION procedure, 7-39
  RESET_COLLECTION_CHANGED
    procedure, 7-40
  RESET_COLLECTION_CHANGED_ALL
    procedure, 7-40
  SORT_MEMBERS procedure, 7-41
  TRUNCATE_COLLECTION procedure, 7-42
  UPDATE_MEMBER procedure, 7-43
APEX_COLLECTION (continued)
  UPDATE_MEMBER_ATTRIBUTE procedure
    signature 1, 7-46
  UPDATE_MEMBER_ATTRIBUTE procedure
    signature 2, 7-48
  UPDATE_MEMBER_ATTRIBUTE procedure
    signature 3, 7-49
  UPDATE_MEMBER_ATTRIBUTE procedure
    signature 4, 7-50
  UPDATE_MEMBER_ATTRIBUTE procedure
    signature 5, 7-52
  UPDATE_MEMBER_ATTRIBUTE procedure
    signature 6, 7-53
  UPDATE_MEMBERS procedure, 7-44
APEX_CREDENTIAL, 8-1
  CLEAR_TOKENS Procedure, 8-1
  SET_PERSISTENT_CREDENTIALS
    Procedure Signature 1, 8-2
  SET_PERSISTENT_CREDENTIALS
    Procedure Signature 2, 8-2
  SET_PERSISTENT_TOKEN Procedure, 8-3
  SET_SESSION_CREDENTIALS Procedure
    Signature 1, 8-4
  SET_SESSION_CREDENTIALS Procedure
    Signature 2, 8-5
  SET_SESSION_TOKEN Procedure, 8-5
APEX_CSS, 9-1
  ADD_3RD_PARTY_LIBRARY_FILE
    procedure, 9-1
  ADD_FILE procedure, 9-2
APEX_CUSTOM_AUTH, 10-1
  APPLICATION_PAGE_ITEM_EXISTS
    function, 10-1
  CURRENT_PAGE_IS_PUBLIC function, 10-2
  DEFINE_USER_SESSION procedure, 10-3
  GET_COOKIE_PROPS, 10-3
  GET_LDAP_PROPS, 10-4
  GET_NEXT_SESSION_ID function, 10-5
  GET_SECURITY_GROUP_ID function, 10-6
  GET_SESSION_ID function, 10-6
  GET_SESSION_ID_FROM_COOKIE, 10-7
  GET_USER function, 10-7
  GET_USERNAME, 10-8
  IS_SESSION_VALID, 10-8
  LOGIN
    Login API, 10-9
  LOGOUT, 10-10
  POST_LOGIN, 10-10
  SESSION_ID_EXISTS function, 10-11
  SET_SESSION_ID procedure, 10-12
  SET_SESSION_ID_TO_NEXT_VALUE
    procedure, 10-12
  SET_USER procedure, 10-13
APEX_DATA_PARSER, 11-1
- data types, 11-1
- Discover, 11-2
- GET_COLUMNS, 11-4
- GET_FILE_PROFILE, 11-5
- GET_FILE_TYPE, 11-8
- GET_XLSX_WORKSHEETS, 11-8
- Global Constants, 11-1
- JSON_TO_PROFILE, 11-9
- PARSE, 11-10

APEX_DEBUG, 12-1
- Constants, 12-2
- DISABLE procedure, 12-2
- DISABLE_DBMS_OUTPUT procedure, 12-3
- ENABLE procedure, 12-3
- ENABLE_DBMS_OUTPUT procedure, 12-6
- ENTER procedure, 12-4
- ERROR procedure, 12-7
- INFO procedure, 12-8
- LOG_DBMS_OUTPUT procedure, 12-9
- LOG_LONG_MESSAGE procedure, 12-10
- LOG_MESSAGE procedure, 12-11
- LOG_PAGE_SESSION_STATE procedure, 12-12
- MESSAGE procedure, 12-13
- REMOVE_DEBUG_BY_AGE procedure, 12-14
- REMOVE_DEBUG_BY_APP procedure, 12-15
- REMOVE_DEBUG_BY_VIEW procedure, 12-16
- REMOVE_SESSION_MESSAGES procedure, 12-16
- TOCHAR function, 12-17
- TRACE procedure, 12-17
- WARN procedure, 12-18

APEX_ERROR, 14-1
- ADD_ERROR Procedure Signature 1, 14-5
- ADD_ERROR Procedure Signature 2, 14-6
- ADD_ERROR Procedure Signature 3, 14-7
- ADD_ERROR Procedure Signature 4, 14-8
- ADD_ERROR Procedure Signature 5, 14-10
- APEX_ERROR.HAVE_ERRORS_OCCURRED Function, 14-11
- AUTO_SET_ASSOCIATED_ITEM Procedure, 14-12
  - example
    - error handling function, 14-3
    - EXTRACT_CONSTRAINT_NAME Function, 14-12
    - GET_FIRST_ORA_ERROR_TEXT Procedure, 14-13
- HTML function, 13-1
- INT_ERROR_RESULT Function, 14-13

APEX_ESCAPE, 13-1
- constants, 13-1
- HTML_ATTRIBUTE Function, 13-3
- HTML_TRUNC Function, 13-3
- HTML_WHITE_LIST Function, 13-5
- JS_LITERAL Function, 13-3
- JSON Function, 13-6
- LDAP_DN Function, 13-7
- LDAP_SEARCH_FILTER Function, 13-8
- NOOP Function, 13-9
- REEXP Function, 13-10
- SET_HTML_ESCAPING_MODE Procedure, 13-10

- data types, 15-5
- Global Constants, 15-3

APEX_EXPORT, 16-1
- GET_APPLICATION Function, 16-1
- GET_FEEDBACK Function, 16-3
- GET_WORKSPACE Function, 16-4
- GET_WORKSPACE_FILES Function, 16-3

APEX_INSTANCE_ADMIN
- ADD_SCHEMA procedure, 17-9
- ADD_WORKSPACE procedure, 17-9
- CREATE_SCHEMA_EXCEPTION Procedure, 17-10
- DB_SIGNATURE function, 17-11
- FREE_WORKSPACE_APP_IDS procedure, 17-12
- GET_PARAMETER function, 17-12
- GET_SCHEMAS function, 17-13
- IS_DB_SIGNATURE_VALID function, 17-14
- parameter values, 17-2
- REMOVE_APPLICATION procedure, 17-15
- REMOVE_SAVED_REPORT procedure, 17-16
- REMOVE_SAVED_REPORTS procedure, 17-16
- REMOVE_SCHEMA procedure, 17-17
- REMOVE_SCHEMA_EXCEPTION Procedure, 17-17
- REMOVE_SCHEMA_EXCEPTIONS Procedure, 17-18
- REMOVE_SUBSCRIPTION Procedure, 17-19
- REMOVE_WORKSPACE procedure, 17-20
- REMOVE_WORKSPACE_EXCEPTIONS Procedure, 17-20
- RESERVE_WORKSPACE_APP_IDS procedure, 17-21
- RESTRICT_SCHEMA Procedure, 17-23
APEX_INSTANCE_ADMIN (continued)
- SET_LOG_SWITCH_INTERVAL Procedure, 17-23
- SET_PARAMETER procedure, 17-25
- SET_WORKSPACE_CONSUMER_GROUP Procedure, 17-26
- TRUNCATE_LOG procedure, 17-27
- UNRESTRICT_SCHEMA Procedure, 17-28
APEX_INSTANCE_ADMIN.GET_WORKSPACE_PARAMETER, 17-14
APEX_INSTANCE_ADMIN.SET_WORKSPACE_PARAMETER, 17-24
APEX_IR
- ADD_FILTER procedure signature 1, 18-1
- ADD_FILTER procedure signature 2, 18-3
- CHANGE_REPORT_OWNER Procedure, 18-5
- CHANGE_SUBSCRIPTION_LANG procedure, 18-6
- CLEAR_REPORT procedure signature 1, 18-7
- CLEAR_REPORT procedure signature 2, 18-8
- DELETE_REPORT procedure, 18-9
- DELETE_SUBSCRIPTION procedure, 18-10
- GET_LAST_VIEWED_REPORT_ID function, 18-10
- GET_REPORT function, 18-11
- RESET_REPORT procedure signature 1, 18-12
- RESET_REPORT procedure signature 2, 18-13
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL, 18-6
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL Procedure, 18-4
APEX_ITEM (continued)
- SELECT_LIST_FROM_QUERY_XL function, 19-23
- SWITCH Function, 19-24
- TEXT function, 19-25
- TEXT_FROM_LOV function, 19-27
- TEXT_FROM_LOV_QUERY function, 19-28
- TEXTAREA function, 19-26
APEX_JAVASCRIPT, 20-1
- ADD_ATTRIBUTE function signature 1, 20-2
- ADD_ATTRIBUTE function signature 2, 20-3
- ADD_ATTRIBUTE function signature 3, 20-4
- ADD_ATTRIBUTE function signature 4, 20-5
- ADD INLINE_CODE procedure, 20-5
- ADD_LIBRARY procedure, 20-6
- ADD_ONLOAD_CODE procedure, 20-8
- ADD_REQUIREJS procedure, 20-7
- ADD_REQUIREJS_DEFINE procedure, 20-8
- ADD.VALUE function signature 1, 20-9
- ADD.VALUE function signature 2, 20-10
- ADD.VALUE function signature 3, 20-10
- ADD.VALUE function signature 4, 20-11
- ESCAPE function, 20-11
APEX_JSON, 21-1, 21-41
- CLOSE ALL procedure, 21-4
- CLOSE_ARRAY procedure, 21-5
- CLOSE OBJECT procedure, 21-5
- constants and datatypes, 21-3
- DOES_EXIST function, 21-5
- FIND_PATHS_LIKE function, 21-6
- FLUSH procedure, 21-8
- GET_BOOLEAN function, 21-9
- GET_CLOB function, 21-19
- GET_CLOB_OUTPUT function, 21-10
- GET_COUNT function, 21-10
- GET_DATE function, 21-11
- GET_MEMBERS function, 21-12
- GET_NUMBER function, 21-13
- GET_T_NUMBER function, 21-15
- GET_T_VARCHAR2 function, 21-17
- GET_VALUE function, 21-14
- GET_VARCHAR2 function, 21-18
- INITIALIZE_OUTPUT procedure, 21-20
- OPEN_ARRAY procedure, 21-22
- OPEN_OBJECT procedure, 21-23
- package overview, 21-2
- PARSE Procedure Signature 1, 21-23
- PARSE Procedure Signature 2, 21-24
- STRINGIFEY Function Signature 1, 21-25
- STRINGIFEY Function Signature 2, 21-26
- STRINGIFEY Function Signature 3, 21-26
- STRINGIFEY Function Signature 4, 21-27
- TO_MEMBER_NAME Function, 21-28
- TO_XMLTYPE Function, 21-29
APEX_JSON (continued)
  TO_XMLTYPE_SQL function, 21-30
  WRITE Procedure Signature 1, 21-30
  WRITE Procedure Signature 10, 21-36
  WRITE Procedure Signature 11, 21-36
  WRITE Procedure Signature 12, 21-37
  WRITE Procedure Signature 13, 21-37
  WRITE Procedure Signature 14, 21-38
  WRITE Procedure Signature 15, 21-39
  WRITE Procedure Signature 16, 21-40
  WRITE Procedure Signature 2, 21-31
  WRITE Procedure Signature 3, 21-32
  WRITE Procedure Signature 4, 21-32
  WRITE Procedure Signature 5, 21-33
  WRITE Procedure Signature 6, 21-33
  WRITE Procedure Signature 7, 21-34
  WRITE Procedure Signature 8, 21-34
  WRITE Procedure Signature 9, 21-35
APEX_JSON.INITIALIZE_OUTPUT procedure, 21-21
APEX_JWT, 22-1, 22-4
  DECODE function, 22-3
  ENCODE function, 22-1
  T_TOKEN, 22-1
APEX_LANG, 23-1
  CREATE_LANGUAGE_MAPPING procedure, 23-1
  DELETE_LANGUAGE_MAPPING procedure, 23-2
  EMIT_LANGUAGE_SELECTOR_LIST procedure, 23-4
  LANG function, 23-4
  MESSAGE function, 23-5
  PUBLISH_APPLICATION procedure, 23-7
  SEED_TRANSLATIONS procedure, 23-8
  UPDATE_LANGUAGE_MAPPING procedure, 23-9
  UPDATE_MESSAGE procedure, 23-10
  UPDATE_TRANSLATED_STRING procedure, 23-12
APEX_LDAP, 24-1
  AUTHENTICATE, 24-1
  GET_ALL_USER_ATTRIBUTES, 24-2
  GET_USER_ATTRIBUTES, 24-3
  IS_MEMBER, 24-5
  MEMBER_OF, 24-6
  MEMBER_OF2 Function, 24-7
APEX_MAIL, 25-1
  ADD_ATTACHMENT procedure, 25-2
  PREPARE_TEMPLATE procedure, 25-5
  PUSH_QUEUE procedure, 25-6
  SEND function, 25-9, 25-13
  SEND procedure, 25-7, 25-10
APEX_MAIL_QUEUE, sending email in queue, 25-6
APEX_PAGE, 27-1
  GET_PAGE_MODE function, 27-2
  GET_UI_TYPE function, 27-2
  GET_URL function, 27-3
  Global Constants, 27-1
  IS_DESKTOP_UI function, 27-1
  IS_JQM_SMARTPHONE_UI function, 27-1
  IS_JQM_TABLET_UI function, 27-2
  IS_READ_ONLY function, 27-2
  PURGE_CACHE procedure, 27-3
APEX_PAGE.PURGE_CACHE Procedure, 27-3
APEX_PKG_APP_INSTALL, 26-1
  DEINSTALL procedure signature 1, 26-1
  DEINSTALL procedure signature 2, 26-2
  INSTALL Function signature 1, 26-2, 26-3
  UPGRADE Procedure signature 1, 26-4
  UPGRADE Procedure signature 2, 26-5
APEX_PLUGIN, 28-1
  data types, 28-1
  GET_AJAX_IDENTIFIER function, 28-8
  GET_INPUT_NAME_FOR_PAGE_ITEM function, 28-9
APEX_PLUGIN_UTIL, 29-1, 29-25
  CLEAR_COMPONENT_VALUES procedure, 29-2
  DEBUG_DYNAMIC_ACTION procedure, 29-2
  DEBUG_PAGE_ITEM procedure signature 1, 29-3
  DEBUG_PAGE_ITEM procedure signature 2, 29-3
  DEBUG_PROCESS procedure, 29-4
  DEBUG_REGION procedure signature 1, 29-5
  DEBUG_REGION procedure signature 2, 29-5
  ESCAPE function, 29-6
  EXECUTE_PLSQL_CODE procedure, 29-7
  GET_ATTRIBUTE_AS_NUMBER function, 29-7
  GET_DATA function signature 1, 29-8
  GET_DATA Function Signature 2, 29-10
  GET_DATA2 function signature 1, 29-12
  GET_DATA2 function signature 2, 29-15
  GET_DISPLAY_DATA function signature 1, 29-17
  GET_DISPLAY_DATA function signature 2, 29-19
  GET_DISPLAY_DATA function signature 2, 29-19
  GET_ELEMENT_ATTRIBUTES function, 29-21
  GET_PLSQL_EXPRESSION_RESULT function, 29-22
  GET_PLSQL_FUNCTION_RESULT function, 29-23
  GET_POSITION_IN_LIST function, 29-23
APEX_PLUGIN_UTIL (continued)
  GET_SEARCH_STRING function, 29-24
  IS_EQUAL function, 29-26
  PAGE_ITEM_NAMES_TO_JQUERY function, 29-27
  PRINT_DISPLAY_ONLY procedure, 29-28
  PRINT_ESCAPED_VALUE procedure, 29-29
  PRINT_HIDDEN_IF_READONLY procedure, 29-30
  PRINT_JSON_HTTP_HEADER procedure, 29-31
  PRINT_LOV_AS_JSON procedure, 29-31
  PRINT_OPTION procedure, 29-32
  REPLACE_SUBSTITUTIONS function, 29-33
  SET_COMPONENT_VALUES procedure, 29-34

APEX_REGION
  IS_READ_ONLY function, 30-1
  PURGE_CACHE procedure, 30-1
APEX_SESSION, 31-1
  Attach_Procedure, 31-5
  CREATE_SESSION_Procedure, 31-3
  DELETE_SESSION_Procedure, 31-4
  Detach_Procedure, 31-6
  SET_DEBUG_Procedure, 31-1
  SET_TRACE_Procedure, 31-2
APEX_SPATIAL, 32-1
  CHANGE_GEOM_METADATA Procedure, 32-1
  CIRCLE_POLYGON Function, 32-3
  Data Types, 32-1
  DELETE_GEOM_METADATA Procedure, 32-3
  INSERT_GEOM_METADATA Procedure, 32-4
  INSERT_GEOM_METADATA_LONLAT Procedure, 32-5
  POINT Function, 32-6
  RECTANGLE Function, 32-7
APEX_STRING, 33-1
  FORMAT Function, 33-1
  GET_INITIALIZS Function, 33-3
  GET_SEARCHABLE_PHRASES Function, 33-4
  GREP Function signature 1, 33-5
  GREP Function signature 2, 33-6
  GREP Function signature 3, 33-6
  JOIN Function signature 1, 33-8
  JOIN Function signature 2, 33-9
  JOIN_CLOB, 33-7
  NEXT_CHUNK Function, 33-9
  PLIST_DELETE Procedure, 33-10
  PLIST_GET Function, 33-11
  PLIST_PUSH Procedure, 33-12
  PLIST_PUT Function, 33-13
APEX_STRING (continued)
  PUSH procedure signature 1, 33-13
  PUSH procedure signature 2, 33-14
  PUSH procedure signature 3, 33-15
  PUSH procedure signature 4, 33-15
  SHUFFLE Function, 33-16
  SHUFFLE Procedure, 33-17
  SPLIT Function signature 1, 33-17
  SPLIT Function signature 2, 33-18
  SPLIT_NUMBERS Function, 33-19
  STRING_TO_TABLE, 33-19
  TABLE_TO_STRING Function, 33-20
APEX_THEME, 34-1
  CLEAR_ALL_USERS_STYLE Procedure, 34-1
  CLEAR_USER_STYLE Procedure, 34-2
  DISABLE_USER_STYLE Procedure, 34-2
  ENABLE_USER_STYLE Procedure, 34-3
  GET_USER_STYLE Function, 34-4
  SET_CURRENT_STYLE Procedure, 34-5
  SET_SESSION_STYLE Procedure, 34-6
  SET_SESSION_STYLE_CSS Procedure, 34-7
  SET_USER_STYLE Procedure, 34-8
APEX_UI_DEFAULT_UPDATE, 35-1
  ADD_AD_COLUMN procedure, 35-2
  ADD_AD_SYNONYM procedure, 35-3
  DEL_AD_COLUMN procedure, 35-4
  DEL_AD_SYNONYM procedure, 35-5
  DEL_COLUMN procedure, 35-5
  DEL_GROUP procedure, 35-6
  DEL_TABLE procedure, 35-7
  SYNCH_TABLE procedure, 35-7
  UPD_AD_COLUMN procedure, 35-8
  UPD_AD_SYNONYM procedure, 35-9
  UPD_COLUMN procedure, 35-10
  UPD_DISPLAY_IN_FORM procedure, 35-12
  UPD_DISPLAY_IN_REPORT procedure, 35-13
  UPD_FORM_REGION_TITLE procedure, 35-13
  UPD_GROUP procedure, 35-14
  UPD_ITEM_DISPLAY_HEIGHT procedure, 35-15
  UPD_ITEM_DISPLAY_WIDTH procedure, 35-16
  UPD_ITEM_FORMAT_MASK procedure, 35-16
  UPD_ITEM_HELP procedure, 35-17
  UPD_ITEM_LABEL procedure, 35-18
  UPD_ITEM_FORMAT_MASK procedure, 35-18
  UPD_REPORT_ALIGNMENT procedure, 35-19
APEX_UI_DEFAULT_UPDATE (continued)
UPD_REPORT_REGION_TITLE procedure, 35-20
UPD_TABLE procedure, 35-20
APEX_UTIL, 36-1
CACHE_GET_DATE_OF_PAGE_CACHE function, 36-5, 36-6
CACHE_PURGE_BY_APPLICATION procedure, 36-6
CACHE_PURGE_BY_PAGE procedure, 36-7
CACHE_PURGE_STALE procedure, 36-8
CHANGE_CURRENT_USER_PW procedure, 36-8
CHANGE_PASSWORD_ON_FIRST_USE function, 36-9
CLEAR_APP_CACHE procedure, 36-11
CLEAR_PAGE_CACHE procedure, 36-11
CLEAR_USER_CACHE procedure, 36-12
COUNT_CLICK procedure, 36-12
CREATE_USER procedure, 36-13
CREATE_USER_GROUP procedure, 36-17
CURRENT_USER_IN_GROUP function, 36-18
CUSTOM_CALENDAR procedure, 36-18
DELETE_USER_GROUP procedure signature 1, 36-19
DELETE_USER_GROUP procedure signature 2, 36-20
DOWNLOAD_PRINT_DOCUMENT procedure signature 1, 36-20
DOWNLOAD_PRINT_DOCUMENT procedure signature 2, 36-21
DOWNLOAD_PRINT_DOCUMENT procedure signature 3, 36-23
DOWNLOAD_PRINT_DOCUMENT procedure signature 4, 36-24
EDIT_USER procedure, 36-25
END_USER_ACCOUNT_DAYS_LEFT function, 36-30
EXPIRE_END_USER_ACCOUNT procedure, 36-30
EXPIRE_WORKSPACE_ACCOUNT procedure, 36-31
EXPORT_USERS procedure, 36-32
FETCH_APP_ITEM function, 36-33
FETCH_USER procedure signature 1, 36-34
FETCH_USER procedure signature 2, 36-36
FETCH_USER procedure signature 3, 36-38
FIND_SECURITY_GROUP_ID function, 36-41
FIND_WORKSPACE function, 36-42
GET_ACCOUNT_LOCKED_STATUS function, 36-43
APEX_UTIL (continued)
GET_APPLICATION_STATUS function, 36-43
GET_ATTRIBUTE function, 36-44
GET_AUTHENTICATION_RESULT function, 36-45
GET_BLOB_FILE_SRC function, 36-46
GET_BUILD_OPTION_STATUS function signature 1, 36-47
GET_BUILD_OPTION_STATUS function signature 2, 36-48
GET_CURRENT_USER_ID function, 36-49
GET_DEFAULT_SCHEMA function, 36-49
GET_EDITION function, 36-50
GET_EMAIL function, 36-50
GET_FEEDBACK_FOLLOW_UP function, 36-51
GET_FILE procedure, 36-52
GET_FILE_ID function, 36-53
GET_FIRST_NAME function, 36-54
GET_GLOBAL_NOTIFICATION function, 36-54
GET_GROUP_ID function, 36-56
GET_GROUP_NAME function, 36-56
GET_GROUPS_USER_BELONGS_TO function, 36-55
GET_HASH function, 36-57
GET_HIGH_CONTRAST_MODE_TOGGLE function, 36-58
GET_LAST_NAME function, 36-59
GET_NUMERIC_SESSION_STATE function, 36-60
GET_PREFERENCE function, 36-61
GET_PRINT_DOCUMENT function signature 1, 36-62
GET_PRINT_DOCUMENT function signature 2, 36-62
GET_PRINT_DOCUMENT function signature 3, 36-63
GET_PRINT_DOCUMENT function signature 4, 36-64
GET_SESSION_LANG function, 36-66
GET_SESSION_STATE function, 36-66
GET_SESSION_TERRITORY function, 36-67
GET_SESSION_TIME_ZONE function, 36-68
GET_SINCE function, 36-68
GET_SUPPORTING_OBJECT_SCRIPT function, 36-70
GET_SUPPORTING_OBJECT_SCRIPT procedure, 36-71
GET_USER_ID function, 36-72
APEXUTIL (continued)

GET_USER_ROLES function, 36-72
GET_USERNAME function, 36-73
HOST_URL function, 36-74
IR_CLEAR procedure, 36-76
IR_DELETE_REPORT procedure, 36-77
IR_DELETE_SUBSCRIPTION procedure, 36-78
IR_FILTER procedure, 36-79
IR_RESET procedure, 36-80
IS_HIGH_CONTRAST_SESSION function, 36-81
IS_HIGH_CONTRAST_SESSION_YN function, 36-82
IS_LOGIN_PASSWORD_VALID function, 36-82
IS_SCREEN_READER_SESSION function, 36-83
IS_SCREEN_READER_SESSION_YN function, 36-84
IS_USERNAME_UNIQUE function, 36-84
KEYVAL_NUM function, 36-85
KEYVAL_VC2 function, 36-85
LOCK_ACCOUNT procedure, 36-86, 36-137
PASSWORD_FIRST_USE_OCCURRED function, 36-87
PREPARE_URL function, 36-88
PUBLIC_CHECK_AUTHORIZATION function, 36-90
PURGE_REGIONS_BY_APP procedure, 36-90
PURGE_REGIONS_BY_NAME procedure, 36-91
PURGE_REGIONS_BY_PAGE procedure, 36-92
REDIRECT_URL procedure, 36-92
REMOVE_PREFERENCE procedure, 36-93
REMOVE_SORT_PREFERENCES procedure, 36-94
REMOVE_USER procedure, 36-95
RESET_AUTHORIZATIONS procedure, 36-95
RESET_PASSWORD Procedure, 36-96
RESET_PW procedure, 36-97
SAVEKEY_NUM function, 36-98
SAVEKEY_VC2 function, 36-99
SET_APP_BUILD_STATUS Procedure, 36-99
SET_APPLICATION_STATUS procedure, 36-100
SET_ATTRIBUTE procedure, 36-102
SET_AUTHENTICATION_RESULT procedure, 36-103
SET_BUILD_OPTION_STATUS procedure, 36-104
APEXUTIL (continued)

SET_CURRENT_THEME_STYLE procedure, 36-105
SET_CUSTOM_AUTH_STATUS procedure, 36-107
SET_EDITION procedure, 36-108
SET_EMAIL procedure, 36-108
SET_FIRST_NAME procedure, 36-109
SET_GLOBAL_NOTIFICATION procedure, 36-110
SET_GROUP_GROUP_GRANTS Procedure, 36-111
SET_GROUP_USER_GRANTS Procedure, 36-112
SET_LAST_NAME procedure, 36-112
SET_PARSING_SCHEMA_FOR_REQUEST procedure, 36-113
SET_PREFERENCE procedure, 36-114
SET_SECURITY_GROUP_ID procedure, 36-115, 36-117
SET_SECURITY_HIGH_CONTRAST_OFF procedure, 36-116
SET_SECURITY_HIGH_CONTRAST_ON procedure, 36-116
SET_SESSION_MAX_IDLE_SECONDS procedure, 36-118
SET_SESSION_SCREEN_READER_OFF procedure, 36-119
SET_SESSION_SCREEN_READER_ON procedure, 36-120
SET_SESSION_STATE procedure, 36-117, 36-120
SET_SESSION_TERRITORY procedure, 36-121
SET_SESSION_TIME_ZONE procedure, 36-122
SET_USERNAME procedure, 36-122
SET_WORKSPACEProcedure, 36-123
SHOW_HIGH_CONTRAST_MODE_TOGGLE procedure, 36-124
SHOW_SCREEN_READER_MODE_TOGGLE procedure, 36-125
STRING_TO_TABLE function, 36-126
STRONG_PASSWORD_CHECK procedure, 36-127
STRONG_PASSWORD_VALIDATION function, 36-130
SUBMIT_FEEDBACK procedure, 36-131
SUBMIT_FEEDBACK_FOLLOWUP procedure, 36-133
TABLE_TO_STRING function, 36-134
UNEXPIRE_END_USER_ACCOUNT procedure, 36-135
UNEXPIRE_WORKSPACE_ACCOUNT procedure, 36-136
APEX_UTIL (continued)
  URL_ENCODE function, 36-138
  WORKSPACE_ACCOUNT_DAYS_LEFT function, 36-139
APEX_WEB_SERVICE, 37-1
  About the APEX_WEB_SERVICE API, 37-1
  BLOB2CLOBBASE64 function, 37-5
  CLOBBASE642BLOB function, 37-6
  Invoking a RESTful Style Web Service, 37-3
  Invoking a SOAP Style Web Service, 37-2
  MAKE_REQUEST function, 37-9
  MAKE_REQUEST procedure, 37-7
  MAKE_REST_REQUEST function, 37-10,
  37-12
  OAUTH_AUTHENTICATE function, 37-14
  OAUTH_GET_LAST_TOKEN function, 37-15
  OAUTH_SET_TOKEN function, 37-15
  PARSE_RESPONSE function, 37-16
  PARSE_RESPONSE_CLOB function, 37-17
  PARSE_XML function, 37-17
  PARSE_XML_CLOB function, 37-19
  Retrieving Cookies and HTTP Headers, 37-4
  Setting Cookies and HTTP Headers, 37-5
APEX_ZIP, 38-1
  ADD_FILE procedure, 38-1
  Data Types, 38-1
  FINISH procedure, 38-2
  GET_FILE_CONTENT function, 38-3
  GET_FILES function, 38-3
APIs
  APEX_ACL, 2-1
  APEX_APP_SETTING, 6-1
  APEX_APPLICATION, 1-1
  APEX_APPLICATION_INSTALL, 3-1
  APEX_AUTHORIZATION, 5-1
  APEX_COLLECTION, 7-2
  APEX_CREDENTIAL, 8-1
  APEX_CSS, 9-1
  APEX_CUSTOM_AUTH, 10-1
  APEX_DATA_PARSER, 11-1
  APEX_DEBUG, 12-1
  APEX_ESCAPE, 13-1
  APEX_EXEC, 15-1
  APEX_ITEM, 19-1
  APEX_JAVASCRIPT, 20-1
  APEX_JSON, 21-1
  APEX_JWT, 22-1
  APEX_LANG, 23-1
  APEX_LDAP, 24-1
  APEX_MAIL, 25-1
  APEX_PAGE, 27-1
  APEX_PKG_APP_INSTALL, 26-1
  APEX_PLUGIN, 28-1
  APEX_PLUGIN_UTIL, 29-1
  APEX_SPATIAL, 32-1

APIs (continued)
  APEX_STRING, 33-1
  APEX_UI_DEFAULT_UPDATE, 35-1
  APEX_UTIL, 36-1
  APEX_WEB_SERVICE, 37-1
  APEX_ZIP, 38-1
  application
    sending messages in APEX_MAIL_QUEUE, 25-6
    sending outbound email, 25-10
    sending outbound email as attachment, 25-2
  attribute values
    setting, 36-102
  authenticated user
    create user group, 36-17
    delete user group, 36-19, 36-20
  authentication, scheme session cookies, 10-3

C
  check box, creating, 19-1
  CLEAR_DML_ROWS Procedure, 15-17
  clicks, counting, 36-12
  CLOSE Procedure, 15-17
  collections
    accessing, 7-5
    APEX_COLLECTION_API, 7-2
    clearing session state, 7-10
    creating, 7-3
    deleting members, 7-8
    determining status, 7-11
    merging, 7-6
    truncating, 7-6
    updating members, 7-8
  COPY_DATA Procedure, 15-18

E
  email
    sending as an attachment, 25-2
    sending messages in APEX_MAIL_QUEUE, 25-6
    sending outbound, 25-10
  EXCECUTE_DML Procedure, 15-20
  EXECUTE_PLSQL Procedure, 15-20
  EXECUTE_REMOTE_PLSQL Procedure, 15-22
  EXECUTE_WEB_SOURCE Procedure, 15-23
  export file, of workspace, 36-32

F
  F01, 1-2
  file repository
    downloading files, 36-52
file repository (continued)
  obtaining primary key, 36-53

G
GET Functions, 15-24, 15-28
GET_COLUMN Function, 15-26
GET_COLUMN_COUNT Function, 15-26
GET_COLUMN_POSITION Function, 15-26
GET_DATA_TYPE Functions, 15-27
GET_DML_STATUS_CODE, 15-30
GET_DML_STATUS_MESSAGE, 15-31
GET_PARAMETER Function, 15-31
GET_ROW_VERSION_CHECKSUM, 15-33
GET_TOTAL_ROW_COUNT Function, 15-33
GET_VALUE_AS_VARCHAR2 function, 29-25

H
HAS_ERROR Function, 15-33

I
Individual REST Services
  Application-Level Statistics, 40-9
  Fetch Instance-Level Statistics, 40-3
  Fetch Workspace-Level Statistics, 40-6
  Instance Overview, 40-11
  REST Service Version Information, 40-13
IS_REMOTE_SQL_AUTH_VALID Function, 15-34

L
LDAP attributes, obtaining, 10-4

N
NEXT_ROW Function, 15-35
NUMBER, 25-13

O
OPEN_LOCAL_DML_CONTEXT Function, 15-35
OPEN_QUERY_CONTEXT Function, 15-45
OPEN_QUERY_CONTEXT Procedure, 15-44
OPEN_REMOTE_DML_CONTEXT Function, 15-39
OPEN_REMOTE_SQL_QUERY Function, 15-42
OPEN_WEB_SOURCE_DML_CONTEXT Function, 15-48
OPEN_WEB_SOURCE_QUERY Function, 15-51

P
password
  changing, 36-8
  resetting and emailing, 36-97
PURGE_WEB_SOURCE_CACHE Procedure, 15-53

R
radio group, generate, 19-16
REST Administration, 40-1
REST Administration interface
  Authentication, 40-1
REST Services
  Individual REST Services, 40-3

S
session cookies, 10-3
session state
  fetching for current application, 36-33
  removing for current page, 36-11
  removing for current session, 36-11
  setting, 36-117, 36-120
SET_NULL Procedure, 15-53
SET_ROW_VERSION_CHECKSUM Procedure, 15-58
SET_VALUE Procedure, 15-54
SET_VALUES Procedure, 15-58
special characters, encoding, 36-138

U
user
  get e-mail address, 36-50
  remove preference, 36-93
user account
  altering, 36-25
  creating new, 36-13
  fetching, 36-34, 36-36, 36-38
  removing, 36-95
  update email address, 36-108
  updating FIRST_NAME, 36-109
  updating LAST_NAME value, 36-112
  updating USER_NAME value, 36-122

V
VALIDATE Procedure, 22-4
variables, global, 1-1
workspace
  export file, 36-32

workspace (continued)
  numeric security group ID, 36-41
  WRITE Procedure Signature 17, 21-41
  WRITE Procedure Signature 18, 21-41