

Oracle® APEX

API Reference



Release 24.2
G12909-02
January 2025



Copyright © 2003, 2025, Oracle and/or its affiliates.

Primary Author: John Godfrey

Contributors: Terri Jennings, Christina Cho, Hilary Farrell, Sharon Kennedy, Christian Neumueller, Anthony Raynor, Marc Sewitz, John Snyders, Jason Straub, Vladislav Uvarov, Patrick Wolf, Stefan Dobre, Ottmar Gobrecht, Ananya Chatterjee

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xxxvii
Documentation Accessibility	xxxvii
Diversity and Inclusion	xxxvii
Related Documents	xxxviii
Conventions	xxxviii

1 Changes in Release 24.2 for Oracle APEX API Reference

2 APEX_ACL

2.1 ADD_USER_ROLE Procedure Signature 1	2-1
2.2 ADD_USER_ROLE Procedure Signature 2	2-2
2.3 HAS_USER_ANY_ROLES Function	2-2
2.4 HAS_USER_ROLE Function	2-3
2.5 IS_ROLE_REMOVED_FROM_USER Function	2-4
2.6 REMOVE_USER_ROLE Procedure Signature 1	2-5
2.7 REMOVE_USER_ROLE Procedure Signature 2	2-6
2.8 REPLACE_USER_ROLES Procedure Signature 1	2-6
2.9 REPLACE_USER_ROLES Procedure Signature 2	2-7
2.10 REMOVE_ALL_USER_ROLES Procedure	2-8

3 APEX_AI

3.1 Constants	3-1
3.2 Data Types	3-1
3.3 CHAT Function Signature 1	3-1
3.4 CHAT Function Signature 2	3-3
3.5 GENERATE Function Signature 1	3-3
3.6 GENERATE Function Signature 2	3-4
3.7 GET_VECTOR_EMBEDDINGS Function Signature 1	3-5
3.8 GET_VECTOR_EMBEDDINGS Function Signature 2	3-6
3.9 GET_VECTOR_EMBEDDINGS Function Signature 3	3-6

3.10	IS_ENABLED Function	3-7
3.11	IS_USER_CONSENT_NEEDED Function	3-7
3.12	REVOKE_USER_CONSENT Procedure	3-8
3.13	REVOKE_USER_CONSENT_FOR_ALL Procedure	3-8
3.14	SET_USER_CONSENT Procedure	3-9
4	APEX_APP_OBJECT_DEPENDENCY	
4.1	Constants	4-1
4.2	CLEAR_CACHE Procedure	4-1
4.3	SCAN Procedure	4-2
5	APEX_APP_SETTING	
5.1	GET_VALUE Function	5-1
5.2	SET_VALUE Procedure	5-1
6	APEX_APPLICATION	
6.1	Working with G_Fnn Arrays (Legacy)	6-1
6.2	Global Variables	6-3
6.3	HELP Procedure	6-3
6.4	STOP_APEX_ENGINE Procedure	6-5
7	APEX_APPLICATION_ADMIN	
7.1	Constants and Data Types	7-2
7.2	GET_APPLICATION_ALIAS Function	7-2
7.3	GET_APPLICATION_NAME Function	7-3
7.4	GET_APPLICATION_STATUS Function	7-4
7.5	GET_APPLICATION_VERSION Function	7-4
7.6	GET_AUTHENTICATION_SCHEME Function	7-5
7.7	GET_BUILD_OPTION_STATUS Function Signature 1	7-5
7.8	GET_BUILD_OPTION_STATUS Function Signature 2	7-6
7.9	GET_BUILD_STATUS Function	7-7
7.10	GET_GLOBAL_NOTIFICATION Function	7-7
7.11	GET_FILE_STORAGE Function	7-8
7.12	GET_IMAGE_PREFIX Function	7-8
7.13	GET_MAX_SCHEDULER_JOBS Function	7-9
7.14	GET_NO_PROXY_DOMAINS Function	7-10
7.15	GET_PARSING_SCHEMA Function	7-10
7.16	GET_PASS_ECID Function	7-11

7.17	GET_PROXY_SERVER Function	7-11
7.18	SET_APPLICATION_ALIAS Procedure	7-12
7.19	SET_APPLICATION_NAME Procedure	7-13
7.20	SET_APPLICATION_STATUS Procedure	7-13
7.21	SET_APPLICATION_VERSION Procedure	7-15
7.22	SET_AUTHENTICATION_SCHEME Procedure	7-15
7.23	SET_BUILD_OPTION_STATUS Procedure	7-16
7.24	SET_BUILD_STATUS Procedure	7-17
7.25	SET_FILE_STORAGE Procedure	7-17
7.26	SET_GLOBAL_NOTIFICATION Procedure	7-18
7.27	SET_IMAGE_PREFIX Procedure	7-19
7.28	SET_MAX_SCHEDULER_JOBS Procedure	7-20
7.29	SET_PARSING_SCHEMA Procedure	7-20
7.30	SET_PASS_ECID Procedure	7-21
7.31	SET_PROXY_SERVER Procedure	7-22
7.32	SET_REMOTE_SERVER Procedure	7-22

8 APEX_APPLICATION_INSTALL

8.1	About the APEX_APPLICATION_INSTALL API	8-2
8.2	Attributes Manipulated by APEX_APPLICATION_INSTALL	8-3
8.3	Import Data Types	8-3
8.4	Import Script Examples	8-4
8.5	CLEAR_ALL Procedure	8-6
8.6	GENERATE_APPLICATION_ID Procedure	8-7
8.7	GENERATE_OFFSET Procedure	8-7
8.8	GET_APPLICATION_ALIAS Function	8-8
8.9	GET_APPLICATION_ID Function	8-8
8.10	GET_APPLICATION_NAME Function	8-9
8.11	GET_AUTHENTICATION_SCHEME Function	8-10
8.12	GET_AUTO_INSTALL_SUP_OBJ Function	8-10
8.13	GET_BUILD_STATUS Function	8-11
8.14	GET_IMAGE_PREFIX Function	8-11
8.15	GET_INFO Function	8-12
8.16	GET_KEEP_BACKGROUND_EXECS Function	8-14
8.17	GET_KEEP_SESSIONS Function	8-14
8.18	GET_MAX_SCHEDULER_JOBS Function	8-15
8.19	GET_NO_PROXY_DOMAINS Function	8-15
8.20	GET_OFFSET Function	8-16
8.21	GET_PASS_ECID Function	8-16
8.22	GET_PROXY Function	8-17
8.23	GET_REMOTE_SERVER_AI_ATTRS Function	8-17

8.24	GET_REMOTE_SERVER_AI_HEADERS Function	8-18
8.25	GET_REMOTE_SERVER_AI_MODEL Function	8-19
8.26	GET_REMOTE_SERVER_BASE_URL Function	8-19
8.27	GET_REMOTE_SERVER_DEFAULT_DB Function	8-20
8.28	GET_REMOTE_SERVER_HTTPS_HOST Function	8-21
8.29	GET_REMOTE_SERVER_SQL_MODE Function	8-21
8.30	GET_REST_SOURCE_CATALOG_GROUP Function	8-22
8.31	GET_SCHEMA Function	8-23
8.32	GET_THEME_ID Function	8-23
8.33	GET_WORKSPACE_ID Function	8-24
8.34	INSTALL Procedure	8-24
8.35	REMOVE_APPLICATION Procedure	8-26
8.36	SET_APPLICATION_ALIAS Procedure	8-26
8.37	SET_APPLICATION_ID Procedure	8-27
8.38	SET_APPLICATION_NAME Procedure	8-27
8.39	SET_AUTHENTICATION_SCHEME Procedure	8-28
8.40	SET_AUTO_INSTALL_SUP_OBJ Procedure	8-29
8.41	SET_BUILD_STATUS Function	8-30
8.42	SET_IMAGE_PREFIX Procedure	8-30
8.43	SET_KEEP_BACKGROUND_EXECS Procedure	8-31
8.44	SET_KEEP_SESSIONS Procedure	8-32
8.45	SET_MAX_SCHEDULER_JOBS Procedure	8-33
8.46	SET_OFFSET Procedure	8-33
8.47	SET_PASS_ECID Procedure	8-34
8.48	SET_PROXY Procedure	8-35
8.49	SET_REMOTE_SERVER Procedure	8-36
8.50	SET_REST_SOURCE_CATALOG_GROUP Procedure	8-37
8.51	SET_SCHEMA Procedure	8-38
8.52	SET_THEME_ID Procedure	8-39
8.53	SET_WORKSPACE_ID Procedure	8-39
8.54	SET_WORKSPACE Procedure	8-40
8.55	SUSPEND_BACKGROUND_EXECS Procedure	8-40

9 APEX_APPROVAL (Deprecated)

9.1	Constants and Data Types	9-2
9.2	ADD_TASK_COMMENT Procedure	9-4
9.3	ADD_TASK_POTENTIAL_OWNER Procedure	9-5
9.4	ADD_TO_HISTORY Procedure	9-6
9.5	APPROVE_TASK Procedure	9-7
9.6	CANCEL_TASK Procedure	9-8
9.7	CLAIM_TASK Procedure	9-9

9.8	COMPLETE_TASK Procedure	9-9
9.9	CREATE_TASK Function	9-10
9.10	DELEGATE_TASK Procedure	9-12
9.11	GET_LOV_PRIORITY Function	9-13
9.12	GET_LOV_STATE Function	9-13
9.13	GET_NEXT_PURGE_TIMESTAMP Function	9-14
9.14	GET_TASK_DELEGATES Function	9-15
9.15	GET_TASK_HISTORY Function	9-15
9.16	GET_TASK_PARAMETER_OLD_VALUE Function	9-17
9.17	GET_TASK_PARAMETER_VALUE Function	9-18
9.18	GET_TASK_PRIORITIES Function	9-19
9.19	GET_TASKS Function	9-19
9.20	HANDLE_TASK_DEADLINES Procedure	9-21
9.21	HAS_TASK_PARAM_CHANGED Function	9-22
9.22	IS_ALLOWED Function	9-23
9.23	IS_BUSINESS_ADMIN Function	9-24
9.24	IS_OF_PARTICIPANT_TYPE Function	9-25
9.25	REJECT_TASK Procedure	9-26
9.26	RELEASE_TASK Procedure	9-27
9.27	REMOVE_POTENTIAL_OWNER Procedure	9-27
9.28	RENEW_TASK Function	9-28
9.29	REQUEST_MORE_INFORMATION Procedure	9-29
9.30	SET_INITIATOR_CAN_COMPLETE Procedure	9-30
9.31	SET_TASK_DUE Procedure	9-31
9.32	SET_TASK_PARAMETER_VALUES Procedure	9-32
9.33	SET_TASK_PRIORITY Procedure	9-33
9.34	SUBMIT_INFORMATION Procedure	9-33

10 APEX_AUTHENTICATION

10.1	Constants	10-1
10.2	CALLBACK Procedure	10-1
10.3	CALLBACK2 Procedure	10-4
10.4	GET_CALLBACK_URL Function	10-5
10.5	GET_LOGIN_USERNAME_COOKIE Function	10-5
10.6	IS_AUTHENTICATED Function	10-6
10.7	IS_PUBLIC_USER Function	10-7
10.8	LOGIN Procedure	10-7
10.9	LOGOUT Procedure	10-8
10.10	PERSISTENT_AUTH_ENABLED Function	10-9
10.11	PERSISTENT_COOKIES_ENABLED Function	10-9
10.12	POST_LOGIN Procedure	10-10

10.13	REMOVE_CURRENT_PERSISTENT_AUTH Procedure	10-10
10.14	REMOVE_PERSISTENT_AUTH Procedure	10-11
10.15	SAML_CALLBACK Procedure	10-12
10.16	SAML_METADATA Procedure	10-12
10.17	SEND_LOGIN_USERNAME_COOKIE Procedure	10-13

11 APEX_AUTHORIZATION

11.1	ENABLE_DYNAMIC_GROUPS Procedure	11-1
11.2	ISAUTHORIZED Function	11-2
11.3	RESET_CACHE Procedure	11-3

12 APEX_AUTOMATION

12.1	ABORT Procedure (Deprecated)	12-1
12.2	DISABLE Procedure	12-2
12.3	ENABLE Procedure	12-3
12.4	EXECUTE Procedure Signature 1	12-3
12.5	EXECUTE Procedure Signature 2	12-4
12.6	EXECUTE for Query Context Procedure	12-5
12.7	EXIT Procedure	12-6
12.8	GET_LAST_RUN Function	12-6
12.9	GET_LAST_RUN_TIMESTAMP Function	12-7
12.10	GET_SCHEDULER_JOB_NAME Function	12-8
12.11	IS_RUNNING Function	12-8
12.12	LOG_ERROR Procedure	12-9
12.13	LOG_INFO Procedure	12-10
12.14	LOG_WARN Procedure	12-10
12.15	RESCHEDULE Procedure	12-11
12.16	SKIP_CURRENT_ROW Procedure	12-12
12.17	TERMINATE Procedure	12-12

13 APEX_BACKGROUND_PROCESS

13.1	Constants	13-1
13.2	Data Types	13-2
13.3	ABORT Procedure Signature 1 (Deprecated)	13-3
13.4	ABORT Procedure Signature 2 (Deprecated)	13-3
13.5	GET_CURRENT_EXECUTION Function	13-4
13.6	GET_EXECUTION Function	13-5
13.7	SET_PROGRESS Procedure	13-5
13.8	SET_STATUS Procedure	13-7

13.9	TERMINATE Procedure Signature 1	13-7
13.10	TERMINATE Procedure Signature 2	13-8

14 APEX_BARCODE

14.1	GET_CODE128_PNG Function	14-1
14.2	GET_CODE128_SVG Function	14-2
14.3	GET_EAN8_PNG Function	14-3
14.4	GET_EAN8_SVG Function	14-4
14.5	GET_QRCODE_PNG Function	14-5
14.6	GET_QRCODE_SVG Function	14-6

15 APEX_COLLECTION

15.1	About the APEX_COLLECTION API	15-2
15.1.1	Accessing a Collection	15-2
15.1.2	Determining Collection Status	15-3
15.1.3	Clearing Collection Session State	15-4
15.2	ADD_MEMBER Procedure	15-4
15.3	ADD_MEMBER Function	15-6
15.4	ADD_MEMBERS Procedure	15-7
15.5	COLLECTION_EXISTS Function	15-9
15.6	COLLECTION_HAS_CHANGED Function	15-9
15.7	COLLECTION_MEMBER_COUNT Function	15-10
15.8	CREATE_COLLECTION Procedure	15-10
15.9	CREATE_COLLECTION_FROM_QUERY Procedure	15-11
15.10	CREATE_COLLECTION_FROM_QUERY2 Procedure	15-12
15.11	CREATE_COLLECTION_FROM_QUERY_B Procedure	15-13
15.12	CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version)	15-15
15.13	CREATE_COLLECTION_FROM_QUERYB2 Procedure	15-16
15.14	CREATE_COLLECTION_FROM_QUERYB2 Procedure (No bind version)	15-17
15.15	CREATE_OR_TRUNCATE_COLLECTION Procedure	15-18
15.16	DELETE_ALL_COLLECTIONS Procedure	15-19
15.17	DELETE_ALL_COLLECTIONS_SESSION Procedure	15-19
15.18	DELETE_COLLECTION Procedure	15-20
15.19	DELETE_MEMBER Procedure	15-20
15.20	DELETE_MEMBERS Procedure	15-21
15.21	GET_MEMBER_MD5 Function	15-22
15.22	MERGE_MEMBERS Procedure	15-23
15.23	MOVE_MEMBER_DOWN Procedure	15-25
15.24	MOVE_MEMBER_UP Procedure	15-26
15.25	RESEQUENCE_COLLECTION Procedure	15-27

15.26	RESET_COLLECTION_CHANGED Procedure	15-27
15.27	RESET_COLLECTION_CHANGED_ALL Procedure	15-28
15.28	SORT_MEMBERS Procedure	15-28
15.29	TRUNCATE_COLLECTION Procedure	15-29
15.30	UPDATE_MEMBER Procedure	15-30
15.31	UPDATE_MEMBERS Procedure	15-31
15.32	UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1	15-33
15.33	UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2	15-34
15.34	UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3	15-35
15.35	UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4	15-36
15.36	UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5	15-37
15.37	UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6	15-38

16 APEX_CREDENTIAL

16.1	CLEAR_TOKENS Procedure	16-1
16.2	CREATE_CREDENTIAL Procedure Signature 1	16-2
16.3	CREATE_CREDENTIAL Procedure Signature 2	16-3
16.4	DROP_CREDENTIAL Procedure	16-4
16.5	SET_ALLOWED_URLS Procedure	16-5
16.6	SET_DATABASE_CREDENTIAL Procedure	16-6
16.7	SET_PERSISTENT_CREDENTIALS Procedure Signature 1	16-7
16.8	SET_PERSISTENT_CREDENTIALS Procedure Signature 2	16-8
16.9	SET_PERSISTENT_CREDENTIALS Procedure Signature 3	16-8
16.10	SET_PERSISTENT_TOKEN Procedure	16-9
16.11	SET_SCOPE Procedure	16-10
16.12	SET_SESSION_CREDENTIALS Procedure Signature 1	16-10
16.13	SET_SESSION_CREDENTIALS Procedure Signature 2	16-11
16.14	SET_SESSION_CREDENTIALS Procedure Signature 3	16-12
16.15	SET_SESSION_TOKEN Procedure	16-12

17 APEX_CSS

17.1	ADD Procedure	17-1
17.2	ADD_3RD_PARTY_LIBRARY_FILE Procedure (Deprecated)	17-1
17.3	ADD_FILE Procedure	17-2

18 APEX_CUSTOM_AUTH

18.1	APPLICATION_PAGE_ITEM_EXISTS Function	18-1
18.2	CURRENT_PAGE_IS_PUBLIC Function	18-2
18.3	DEFINE_USER_SESSION Procedure	18-3

18.4	GET_COOKIE_PROPS Procedure	18-3
18.5	GET_LDAP_PROPS Procedure	18-4
18.6	GET_NEXT_SESSION_ID Function	18-5
18.7	GET_SECURITY_GROUP_ID Function	18-6
18.8	GET_SESSION_ID Function	18-6
18.9	GET_SESSION_ID_FROM_COOKIE Function	18-6
18.10	GET_USER Function	18-7
18.11	GET_USERNAME Function	18-7
18.12	IS_SESSION_VALID Function	18-8
18.13	LDAP_DNPREP Function	18-8
18.14	LOGIN Procedure	18-9
18.15	LOGOUT Procedure (Deprecated)	18-10
18.16	POST_LOGIN Procedure	18-10
18.17	SESSION_ID_EXISTS Function	18-11
18.18	SET_SESSION_ID Procedure	18-12
18.19	SET_SESSION_ID_TO_NEXT_VALUE Procedure	18-12
18.20	SET_USER Procedure	18-12

19 APEX_DATA_LOADING

19.1	Data Types	19-1
19.2	GET_FILE_PROFILE Function	19-1
19.3	LOAD_DATA Function Signature 1	19-2
19.4	LOAD_DATA Function Signature 2	19-3

20 APEX_DATA_EXPORT

20.1	Global Constants	20-1
20.2	Data Types	20-3
20.3	ADD_AGGREGATE Procedure	20-5
20.4	ADD_COLUMN Procedure	20-7
20.5	ADD_COLUMN_GROUP Procedure	20-8
20.6	ADD_HIGHLIGHT Procedure	20-10
20.7	DOWNLOAD Procedure	20-11
20.8	EXPORT Function	20-12
20.9	GET_PRINT_CONFIG Procedure	20-14

21 APEX_DATA_INSTALL

21.1	LOAD_SUPPORTING_OBJECT_DATA Procedure	21-1
------	---------------------------------------	------

22 APEX_DATA_PARSER

22.1	Global Constants	22-1
22.2	Data Types	22-1
22.3	ASSERT_FILE_TYPE Function	22-2
22.4	DISCOVER Function	22-3
22.5	GET_COLUMNS Function	22-5
22.6	GET_FILE_PROFILE Function	22-6
22.7	GET_FILE_TYPE Function	22-8
22.8	GET_XLSX_WORKSHEETS Function	22-9
22.9	JSON_TO_PROFILE Function	22-10
22.10	PARSE Function	22-10
22.11	SET_PARSER_FLAGS Procedure	22-15

23 APEX_DEBUG

23.1	Constants	23-2
23.2	DISABLE Procedure	23-2
23.3	DISABLE_DBMS_OUTPUT Procedure	23-3
23.4	ENABLE Procedure	23-3
23.5	ENTER Procedure	23-4
23.6	ENABLE_DBMS_OUTPUT Procedure	23-5
23.7	ERROR Procedure	23-6
23.8	GET_LAST_MESSAGE_ID Function	23-7
23.9	GET_PAGE_VIEW_ID Function	23-8
23.10	INFO Procedure	23-8
23.11	LOG_DBMS_OUTPUT Procedure	23-9
23.12	LOG_LONG_MESSAGE Procedure	23-10
23.13	LOG_MESSAGE Procedure (Deprecated)	23-11
23.14	LOG_PAGE_SESSION_STATE Procedure	23-13
23.15	MESSAGE Procedure	23-13
23.16	REMOVE_DEBUG_BY_AGE Procedure	23-15
23.17	REMOVE_DEBUG_BY_APP Procedure	23-15
23.18	REMOVE_DEBUG_BY_VIEW Procedure	23-16
23.19	REMOVE_SESSION_MESSAGES Procedure	23-16
23.20	TOCHAR Function	23-17
23.21	TRACE Procedure	23-18
23.22	WARN Procedure	23-19

24 APEX_DG_DATA_GEN

24.1	ADD_BLUEPRINT Procedure	24-2
------	-------------------------	------

24.2	ADD_BLUEPRINT_FROM_FILE Procedure	24-3
24.3	ADD_BLUEPRINT_FROM_TABLES Procedure	24-4
24.4	ADD_COLUMN Procedure	24-5
24.5	ADD_DATA_SOURCE Procedure	24-8
24.6	ADD_TABLE Procedure	24-9
24.7	EXPORT_BLUEPRINT Function	24-11
24.8	GENERATE_DATA Procedure Signature 1	24-12
24.9	GENERATE_DATA Procedure Signature 2	24-13
24.10	GENERATE_DATA_INTO_COLLECTION Procedure	24-15
24.11	GET_BLUEPRINT_ID Function	24-16
24.12	GET_BP_TABLE_ID Function	24-17
24.13	GET_EXAMPLE Function	24-18
24.14	GET_WEIGHTED_INLINE_DATA Function	24-18
24.15	IMPORT_BLUEPRINT Procedure	24-19
24.16	PREVIEW_BLUEPRINT Procedure	24-20
24.17	REMOVE_BLUEPRINT Procedure	24-20
24.18	REMOVE_COLUMN Procedure	24-21
24.19	REMOVE_DATA_SOURCE Procedure	24-21
24.20	REMOVE_TABLE Procedure	24-22
24.21	RESEQUENCE_BLUEPRINT Procedure	24-22
24.22	STOP_DATA_GENERATION Procedure	24-23
24.23	UPDATE_BLUEPRINT Procedure	24-24
24.24	UPDATE_COLUMN Procedure	24-24
24.25	UPDATE_DATA_SOURCE Procedure	24-28
24.26	UPDATE_TABLE Procedure	24-29
24.27	VALIDATE_BLUEPRINT Procedure	24-30
24.28	VALIDATE_INSTANCE_SETTING Procedure	24-30

25 APEX_ERROR

25.1	Constants and Attributes Used for Result Types	25-1
25.2	Example of an Error Handling Function	25-3
25.3	ADD_ERROR Procedure Signature 1	25-5
25.4	ADD_ERROR Procedure Signature 2	25-6
25.5	ADD_ERROR Procedure Signature 3	25-7
25.6	ADD_ERROR Procedure Signature 4	25-8
25.7	ADD_ERROR Procedure Signature 5	25-9
25.8	AUTO_SET_ASSOCIATED_ITEM Procedure	25-10
25.9	EXTRACT_CONSTRAINT_NAME Function	25-11
25.10	GET_FIRST_ORA_ERROR_TEXT Function	25-11
25.11	HAVE_ERRORS_OCCURRED Function	25-12

25.12	INIT_ERROR_RESULT Function	25-12
-------	----------------------------	-------

26 APEX_ESCAPE

26.1	Constants	26-2
26.2	CSS_SELECTOR Function	26-2
26.3	CSV Function Signature 1	26-2
26.4	CSV Function Signature 2	26-3
26.5	GET_CSV_ENCLOSED_BY Function	26-4
26.6	GET_CSV_SEPARATED_BY Function	26-5
26.7	HTML Function	26-5
26.8	HTML_ALLOWLIST Function	26-7
26.9	HTML_ALLOWLIST_CLOB Function	26-7
26.10	HTML_ATTRIBUTE Function	26-8
26.11	HTML_ATTRIBUTE_CLOB Function	26-10
26.12	HTML_CLOB Function	26-10
26.13	HTML_TRUNC Function Signature 1	26-12
26.14	HTML_TRUNC Function Signature 2	26-13
26.15	JS_LITERAL Function	26-14
26.16	JS_LITERAL_CLOB Function	26-15
26.17	JSON Function	26-16
26.18	JSON_CLOB Function	26-16
26.19	LDAP_DN Function	26-17
26.20	LDAP_SEARCH_FILTER Function	26-18
26.21	NOOP Function Signature 1	26-19
26.22	NOOP Function Signature 2	26-19
26.23	REGEXP Function	26-20
26.24	SET_CSV_PARAMETERS Procedure	26-20
26.25	SET_HTML_ESCAPING_MODE Procedure	26-21
26.26	STRIPHTML Function Signature 1	26-22
26.27	STRIPHTML Function Signature 2	26-23

27 APEX_EXEC

27.1	Call Sequences for APEX_EXEC	27-3
27.1.1	Querying a Data Source with APEX_EXEC	27-3
27.1.2	Executing a DML on a Data Source with APEX_EXEC	27-4
27.1.3	Executing a Remote Procedure or REST API with APEX_EXEC	27-5
27.2	Global Constants	27-6
27.3	Data Types	27-10
27.4	ADD_COLUMN Procedure	27-14
27.5	ADD_DML_ARRAY_ROW Procedure	27-15

27.6	ADD_DML_ROW Procedure	27-20
27.7	ADD_FILTER Procedures	27-21
27.8	ADD_ORDER_BY Procedure	27-27
27.9	ADD_PARAMETER Procedure	27-28
27.10	CLEAR_DML_ROWS Procedure	27-31
27.11	CLOSE Procedure	27-31
27.12	CLOSE_ARRAY Procedure	27-31
27.13	COLUMN_EXISTS Function	27-32
27.14	COPY_DATA Procedure	27-33
27.15	DESCRIBE_QUERY Function Signature 1	27-35
27.16	DESCRIBE_QUERY Function Signature 2	27-35
27.17	ENQUOTE_LITERAL Function	27-37
27.18	ENQUOTE_NAME Function	27-38
27.19	EXECUTE_DML Procedure	27-38
27.20	EXECUTE_PLSQL Procedure Signature 1	27-39
27.21	EXECUTE_PLSQL Procedure Signature 2	27-40
27.22	EXECUTE_REMOTE_PLSQL Procedure Signature 1	27-41
27.23	EXECUTE_REMOTE_PLSQL Procedure Signature 2	27-42
27.24	EXECUTE_REST_SOURCE Procedure Signature 1	27-42
27.25	EXECUTE_REST_SOURCE Procedure Signature 2	27-44
27.26	EXECUTE_WEB_SOURCE Procedure (Deprecated)	27-44
27.27	GET Function	27-46
27.28	GET_ARRAY_ROW_DML_OPERATION Function	27-48
27.29	GET_ARRAY_ROW_VERSION_CHECKSUM Function	27-49
27.30	GET_COLUMN Function	27-50
27.31	GET_COLUMNS Function	27-50
27.32	GET_COLUMN_COUNT Function	27-50
27.33	GET_COLUMN_POSITION Function	27-51
27.34	GET_DATA_TYPE Function	27-51
27.35	GET_DML_STATUS_CODE Function	27-52
27.36	GET_DML_STATUS_MESSAGE Function	27-53
27.37	GET_PARAMETER Functions	27-53
27.38	GET_ROW_VERSION_CHECKSUM Function	27-54
27.39	GET_TOTAL_ROW_COUNT Function	27-54
27.40	HAS_ERROR Function	27-55
27.41	HAS_MORE_ARRAY_ROWS Function	27-55
27.42	HAS_MORE_ROWS Function	27-56
27.43	IS_GROUP_END Function	27-57
27.44	IS_REMOTE_SQL_AUTH_VALID Function	27-59
27.45	NEXT_ARRAY_ROW Function	27-59
27.46	NEXT_ROW Function	27-60
27.47	OPEN_ARRAY Procedure Signature 1	27-60

27.48	OPEN_ARRAY Procedure Signature 2	27-62
27.49	OPEN_DUALITY_VIEW_DML_CONTEXT Function	27-63
27.50	OPEN_JSON_SOURCE_DML_CONTEXT Function	27-65
27.51	OPEN_LOCAL_DML_CONTEXT Function	27-67
27.52	OPEN_QUERY_CONTEXT Function Signature 1	27-70
27.53	OPEN_QUERY_CONTEXT Function Signature 2	27-74
27.54	OPEN_REMOTE_DML_CONTEXT Function	27-75
27.55	OPEN_REMOTE_SQL_QUERY Function	27-78
27.56	OPEN_REST_SOURCE_DML_CONTEXT Function	27-80
27.57	OPEN_REST_SOURCE_QUERY Function	27-83
27.58	OPEN_WEB_SOURCE_DML_CONTEXT Function (Deprecated)	27-85
27.59	OPEN_WEB_SOURCE_QUERY Function (Deprecated)	27-87
27.60	PURGE_DUALITY_VIEW_CACHE Procedure	27-89
27.61	PURGE_JSON_SOURCE_CACHE Procedure	27-90
27.62	PURGE_REST_SOURCE_CACHE Procedure	27-90
27.63	PURGE_WEB_SOURCE_CACHE Procedure (Deprecated)	27-91
27.64	SET_ARRAY_CURRENT_ROW Procedure	27-92
27.65	SET_ARRAY_ROW_VERSION_CHECKSUM Procedure	27-92
27.66	SET_CURRENT_ROW Procedure	27-93
27.67	SET_NULL Procedure	27-93
27.68	SET_ROW_VERSION_CHECKSUM Procedure	27-94
27.69	SET_VALUE Procedure	27-96
27.70	SET_VALUES Procedure	27-100

28 APEX_EXPORT

28.1	GET_APPLICATION Function	28-1
28.2	GET_WORKSPACE_FILES Function	28-4
28.3	GET_FEEDBACK Function	28-4
28.4	GET_WORKSPACE Function	28-5
28.5	UNZIP Function	28-6
28.6	ZIP Function	28-7

29 APEX_EXTENSION

29.1	ADD_MENU_ENTRY Procedure	29-1
29.2	GET_GRANTOR_WORKSPACE Function	29-2
29.3	REMOVE_MENU_ENTRY Procedure	29-3
29.4	SET_WORKSPACE Procedure Signature 1	29-3
29.5	SET_WORKSPACE Procedure Signature 2	29-4

30 APEX_HTTP

30.1	DOWNLOAD Procedure Signature 1	30-1
30.2	DOWNLOAD Procedure Signature 2	30-2

31 APEX_HUMAN_TASK

31.1	Constants and Data Types	31-2
31.2	ADD_TASK_COMMENT Procedure	31-4
31.3	ADD_TASK_POTENTIAL_OWNER Procedure	31-5
31.4	ADD_TO_HISTORY Procedure	31-5
31.5	APPROVE_TASK Procedure	31-6
31.6	CANCEL_TASK Procedure	31-7
31.7	CLAIM_TASK Procedure	31-7
31.8	COMPLETE_TASK Procedure	31-8
31.9	CREATE_TASK Function	31-9
31.10	DELEGATE_TASK Procedure	31-10
31.11	GET_LOV_PRIORITY Function	31-11
31.12	GET_LOV_STATE Function	31-11
31.13	GET_LOV_TYPE Function	31-12
31.14	GET_NEXT_PURGE_TIMESTAMP Function	31-12
31.15	GET_TASK_DELEGATES Function	31-13
31.16	GET_TASK_HISTORY Function	31-13
31.17	GET_TASK_PARAMETER_OLD_VALUE Function	31-14
31.18	GET_TASK_PARAMETER_VALUE Function	31-14
31.19	GET_TASK_PRIORITIES Function	31-15
31.20	GET_TASKS Function	31-16
31.21	HANDLE_TASK_DEADLINES Procedure	31-18
31.22	HAS_TASK_PARAM_CHANGED Function	31-18
31.23	IS_ALLOWED Function	31-19
31.24	IS_BUSINESS_ADMIN Function	31-20
31.25	IS_OF_PARTICIPANT_TYPE Function	31-20
31.26	REJECT_TASK Procedure	31-21
31.27	RELEASE_TASK Procedure	31-22
31.28	REMOVE_POTENTIAL_OWNER Procedure	31-23
31.29	RENEW_TASK Function	31-23
31.30	REQUEST_MORE_INFORMATION Procedure	31-24
31.31	SET_INITIATOR_CAN_COMPLETE Procedure	31-25
31.32	SET_TASK_DUE Procedure	31-25
31.33	SET_TASK_PARAMETER_VALUES Procedure	31-26
31.34	SET_TASK_PRIORITY Procedure	31-27

32 APEX_INSTANCE_ADMIN

32.1	Available Parameter Values	32-2
32.2	ADD_AUTO_PROV_RESTRICTIONS Procedure	32-14
32.3	ADD_SCHEMA Procedure	32-15
32.4	ADD_WEB_ENTRY_POINT Procedure	32-15
32.5	ADD_WORKSPACE Procedure	32-16
32.6	CREATE_CLOUD_CREDENTIAL Procedure	32-17
32.7	CREATE_OR_UPDATE_ADMIN_USER Procedure	32-18
32.8	CREATE_SCHEMA_EXCEPTION Procedure	32-18
32.9	DB_SIGNATURE Function	32-19
32.10	DROP_CLOUD_CREDENTIAL Procedure	32-20
32.11	FREE_WORKSPACE_APP_IDS Procedure	32-20
32.12	GET_PARAMETER Function	32-21
32.13	GET_SCHEMAS Function	32-21
32.14	GET_WORKSPACE_PARAMETER Procedure	32-22
32.15	GRANT_EXTENSION_WORKSPACE Procedure	32-23
32.16	IS_DB_SIGNATURE_VALID Function	32-24
32.17	REMOVE_APPLICATION Procedure	32-25
32.18	REMOVE_AUTO_PROV_RESTRICTIONS Procedure	32-25
32.19	REMOVE_SAVED_REPORT Procedure	32-26
32.20	REMOVE_SAVED_REPORTS Procedure	32-26
32.21	REMOVE_SCHEMA Procedure	32-27
32.22	REMOVE_SCHEMA_EXCEPTION Procedure	32-27
32.23	REMOVE_SCHEMA_EXCEPTIONS Procedure	32-28
32.24	REMOVE_SUBSCRIPTION Procedure	32-29
32.25	REMOVE_WEB_ENTRY_POINT Procedure	32-30
32.26	REMOVE_WORKSPACE Procedure	32-30
32.27	REMOVE_WORKSPACE_EXCEPTIONS Procedure	32-31
32.28	RESERVE_WORKSPACE_APP_IDS Procedure	32-32
32.29	RESTRICT_SCHEMA Procedure	32-33
32.30	REVOKE_EXTENSION_WORKSPACE Procedure	32-34
32.31	SET_LOG_SWITCH_INTERVAL Procedure	32-34
32.32	SET_PARAMETER Procedure	32-35
32.33	SET_WORKSPACE_CONSUMER_GROUP Procedure	32-36
32.34	SET_WORKSPACE_PARAMETER Procedure	32-36
32.35	TRUNCATE_LOG Procedure	32-38
32.36	UNLOCK_USER Procedure	32-39
32.37	UNRESTRICT_SCHEMA Procedure	32-39

32.38	VALIDATE_EMAIL_CONFIG Procedure	32-40
-------	---------------------------------	-------

33 APEX_IG

33.1	ADD_FILTER Procedure Signature 1	33-1
33.2	ADD_FILTER Procedure Signature 2	33-3
33.3	CHANGE_REPORT_OWNER Procedure	33-5
33.4	CLEAR_REPORT Procedure Signature 1	33-5
33.5	CLEAR_REPORT Procedure Signature 2	33-6
33.6	DELETE_REPORT Procedure	33-7
33.7	GET_LAST_VIEWED_REPORT_ID Function	33-8
33.8	RESET_REPORT Procedure Signature 1	33-9
33.9	RESET_REPORT Procedure Signature 2	33-9

34 APEX_IR

34.1	Constants and Data Types	34-1
34.2	ADD_FILTER Procedure Signature 1	34-2
34.3	ADD_FILTER Procedure Signature 2	34-3
34.4	CHANGE_REPORT_OWNER Procedure	34-5
34.5	CHANGE_SUBSCRIPTION_EMAIL Procedure	34-5
34.6	CHANGE_SUBSCRIPTION_LANG Procedure	34-6
34.7	CLEAR_REPORT Procedure Signature 1	34-7
34.8	CLEAR_REPORT Procedure Signature 2	34-7
34.9	CLONE_REPORT Function	34-8
34.10	DELETE_REPORT Procedure	34-9
34.11	DELETE_SUBSCRIPTION Procedure	34-10
34.12	EXPORT_SAVED_REPORTS Function	34-10
34.13	GET_LAST_VIEWED_REPORT_ID Function	34-11
34.14	GET_REPORT Function (Deprecated)	34-12
34.15	IMPORT_SAVED_REPORTS Procedure	34-13
34.16	RESET_REPORT Procedure Signature 1	34-15
34.17	RESET_REPORT Procedure Signature 2	34-15

35 APEX_ITEM (Legacy)

35.1	CHECKBOX2 Function	35-1
35.2	DATE_POPUP Function	35-3
35.3	DATE_POPUP2 Function	35-4
35.4	DISPLAY_AND_SAVE Function	35-6
35.5	HIDDEN Function	35-6
35.6	MD5_CHECKSUM Function	35-7

35.7	MD5_HIDDEN Function	35-8
35.8	POPUP_FROM_LOV Function	35-9
35.9	POPUP_FROM_QUERY Function	35-11
35.10	POPUPKEY_FROM_LOV Function	35-12
35.11	POPUPKEY_FROM_QUERY Function	35-14
35.12	RADIOGROUP Function	35-15
35.13	SELECT_LIST Function	35-16
35.14	SELECT_LIST_FROM_LOV Function	35-18
35.15	SELECT_LIST_FROM_LOV_XL Function	35-19
35.16	SELECT_LIST_FROM_QUERY Function	35-20
35.17	SELECT_LIST_FROM_QUERY_XL Function	35-21
35.18	SWITCH Function	35-22
35.19	TEXT Function	35-23
35.20	TEXTAREA Function	35-24
35.21	TEXT_FROM_LOV Function	35-25
35.22	TEXT_FROM_LOV_QUERY Function	35-26

36 APEX_JAVASCRIPT

36.1	ADD_3RD_PARTY_LIBRARY_FILE Procedure (Deprecated)	36-1
36.2	ADD_ATTRIBUTE Function Signature 1	36-2
36.3	ADD_ATTRIBUTE Function Signature 2	36-4
36.4	ADD_ATTRIBUTE Function Signature 3	36-4
36.5	ADD_ATTRIBUTE Function Signature 4	36-5
36.6	ADD_INLINE_CODE Procedure	36-5
36.7	ADD_JET Procedure	36-6
36.8	ADD_LIBRARY Procedure	36-6
36.9	ADD_REQUIREJS Procedure	36-8
36.10	ADD_REQUIREJS_DEFINE Procedure	36-8
36.11	ADD_ONLOAD_CODE Procedure	36-8
36.12	ADD_VALUE Function Signature 1	36-9
36.13	ADD_VALUE Function Signature 2	36-10
36.14	ADD_VALUE Function Signature 3	36-10
36.15	ADD_VALUE Function Signature 4	36-11
36.16	Escape Function	36-11

37 APEX_JSON

37.1	APEX_JSON Overview and Examples	37-2
37.2	Constants and Data Types	37-3
37.3	CLOSE_ALL Procedure	37-4
37.4	CLOSE_ARRAY Procedure	37-5

37.5	CLOSE_OBJECT Procedure	37-5
37.6	DOES_EXIST Function	37-5
37.7	FIND_PATHS_LIKE Function	37-6
37.8	FLUSH Procedure	37-7
37.9	FREE_OUTPUT Procedure	37-8
37.10	GET_BOOLEAN Function	37-8
37.11	GET_CLOB Function	37-9
37.12	GET_CLOB_OUTPUT Function	37-10
37.13	GET_COUNT Function	37-11
37.14	GET_DATE Function	37-12
37.15	GET_MEMBERS Function	37-13
37.16	GET_NUMBER Function	37-14
37.17	GET_SDO_GEOMETRY Function	37-15
37.18	GET_T_NUMBER Function	37-16
37.19	GET_T_VARCHAR2 Function	37-17
37.20	GET_VALUE Function	37-19
37.21	GET_VALUE_KIND Function	37-19
37.22	GET_VARCHAR2 Function	37-21
37.23	INITIALIZE_CLOB_OUTPUT Procedure	37-22
37.24	INITIALIZE_OUTPUT Procedure	37-23
37.25	OPEN_ARRAY Procedure	37-23
37.26	OPEN_OBJECT Procedure	37-24
37.27	PARSE Procedure Signature 1	37-25
37.28	PARSE Procedure Signature 2	37-26
37.29	STRINGIFY Function Signature 1	37-26
37.30	STRINGIFY Function Signature 2	37-27
37.31	STRINGIFY Function Signature 3	37-27
37.32	STRINGIFY Function Signature 4	37-28
37.33	STRINGIFY Function Signature 5	37-29
37.34	TO_MEMBER_NAME Function	37-29
37.35	TO_XMLTYPE Function	37-30
37.36	TO_XMLTYPE_SQL Function	37-31
37.37	WRITE Procedure Signature 1	37-32
37.38	WRITE Procedure Signature 2	37-33
37.39	WRITE Procedure Signature 3	37-33
37.40	WRITE Procedure Signature 4	37-33
37.41	WRITE Procedure Signature 5	37-34
37.42	WRITE Procedure Signature 6	37-34
37.43	WRITE Procedure Signature 7	37-35
37.44	WRITE Procedure Signature 8	37-36
37.45	WRITE Procedure Signature 9	37-36
37.46	WRITE Procedure Signature 10	37-37

37.47	WRITE Procedure Signature 11	37-37
37.48	WRITE Procedure Signature 12	37-38
37.49	WRITE Procedure Signature 13	37-38
37.50	WRITE Procedure Signature 14	37-39
37.51	WRITE Procedure Signature 15	37-40
37.52	WRITE Procedure Signature 16	37-40
37.53	WRITE Procedure Signature 17	37-41
37.54	WRITE Procedure Signature 18	37-42
37.55	WRITE Procedure Signature 19	37-42
37.56	WRITE Procedure Signature 20	37-43
37.57	WRITE Procedure Signature 21	37-44
37.58	WRITE_CONTEXT Procedure	37-44

38 APEX_JWT

38.1	t_token Record	38-1
38.2	ENCODE Function	38-1
38.3	DECODE Function	38-3
38.4	VALIDATE Procedure	38-4

39 APEX_LANG

39.1	APPLY_XLIFF_DOCUMENT Procedure	39-1
39.2	CREATE_LANGUAGE_MAPPING Procedure	39-2
39.3	CREATE_MESSAGE Procedure	39-3
39.4	DELETE_LANGUAGE_MAPPING Procedure	39-4
39.5	DELETE_MESSAGE Procedure	39-5
39.6	EMIT_LANGUAGE_SELECTOR_LIST Procedure	39-6
39.7	GET_LANGUAGE_SELECTOR_LIST Function	39-7
39.8	GET_MESSAGE Function	39-7
39.9	GET_XLIFF_DOCUMENT Function	39-8
39.10	LANG Function	39-9
39.11	MESSAGE Function (Deprecated)	39-10
39.12	PUBLISH_APPLICATION Procedure	39-11
39.13	SEED_TRANSLATIONS Procedure	39-12
39.14	UPDATE_LANGUAGE_MAPPING Procedure	39-13
39.15	UPDATE_MESSAGE Procedure Signature 1	39-15
39.16	UPDATE_MESSAGE Procedure Signature 2	39-16
39.17	UPDATE_TRANSLATED_STRING Procedure	39-17

40 APEX_LDAP

40.1	AUTHENTICATE Function	40-1
40.2	GET_ALL_USER_ATTRIBUTES Procedure	40-2
40.3	GET_USER_ATTRIBUTES Procedure	40-3
40.4	IS_MEMBER Function	40-4
40.5	MEMBER_OF Function	40-6
40.6	MEMBER_OF2 Function	40-7
40.7	SEARCH Function	40-8

41 APEX_MAIL

41.1	Configuring Oracle APEX to Send Email	41-2
41.2	ADD_ATTACHMENT Procedure Signature 1	41-2
41.3	ADD_ATTACHMENT Procedure Signature 2	41-4
41.4	GET_IMAGES_URL Function	41-5
41.5	GET_INSTANCE_URL Function	41-6
41.6	PREPARE_TEMPLATE Procedure	41-7
41.7	PUSH_QUEUE Procedure	41-8
41.8	SEND Function Signature 1	41-9
41.9	SEND Function Signature 2	41-12
41.10	SEND Procedure Signature 1	41-14
41.11	SEND Procedure Signature 2	41-17

42 APEX_MARKDOWN

42.1	Constants	42-1
42.2	TO_HTML Function	42-1

43 APEX_PAGE

43.1	Global Constants	43-1
43.2	GET_PAGE_MODE Function	43-1
43.3	GET_UI_TYPE Function (Deprecated)	43-1
43.4	GET_URL Function	43-2
43.5	IS_DESKTOP_UI Function (Deprecated)	43-3
43.6	IS_READ_ONLY Function	43-3
43.7	PURGE_CACHE Procedure	43-4

44 APEX_PLUGIN

44.1	About Configuring Flexible Remote Servers in APEX	44-1
44.2	Constants	44-2

44.3	Data Types	44-3
44.3.1	c_inline_in_notification	44-4
44.3.2	c_inline_with_field	44-4
44.3.3	c_inline_with_field_and_notif	44-4
44.3.4	c_on_error_page	44-4
44.3.5	t_authentication	44-4
44.3.6	t_authentication_ajax_result	44-5
44.3.7	t_authentication_auth_result	44-5
44.3.8	t_authentication_inval_result	44-5
44.3.9	t_authentication_logout_result	44-5
44.3.10	t_authentication_sentry_result	44-5
44.3.11	t_authorization	44-5
44.3.12	t_authorization_exec_result	44-6
44.3.13	t_dynamic_action	44-6
44.3.14	t_dynamic_action_ajax_result	44-7
44.3.15	t_dynamic_action_render_result	44-7
44.3.16	t_escape_mode	44-7
44.3.17	t_item	44-8
44.3.18	t_item_ajax_result	44-9
44.3.19	t_item_meta_data_result	44-9
44.3.20	t_item_render_param	44-10
44.3.21	t_item_render_result	44-10
44.3.22	t_item_validation_result	44-10
44.3.23	t_plugin	44-11
44.3.24	t_plugin_attributes	44-11
44.3.25	t_process	44-12
44.3.26	t_process_exec_result	44-12
44.3.27	t_region	44-13
44.3.28	t_region_ajax_result	44-14
44.3.29	t_region_column	44-14
44.3.30	t_region_columns	44-15
44.3.31	t_region_render_param	44-15
44.3.32	t_region_render_result	44-15
44.3.33	t_remote_server_config	44-15
44.3.34	t_remote_server_info	44-15
44.4	GET_AJAX_IDENTIFIER Function	44-15
44.5	GET_INPUT_NAME_FOR_PAGE_ITEM Function (Deprecated)	44-16

45 APEX_PLUGIN_UTIL

45.1	BUILD_REQUEST_BODY Procedure	45-2
45.2	CLEAR_COMPONENT_VALUES Procedure	45-4

45.3	CURRENT_ROW_CHANGED Function	45-4
45.4	DB_OPERATION_ALLOWED Function	45-6
45.5	DEBUG_DYNAMIC_ACTION Procedure	45-7
45.6	DEBUG_PAGE_ITEM Procedure Signature 1 (Deprecated)	45-7
45.7	DEBUG_PAGE_ITEM Procedure Signature 2 (Deprecated)	45-8
45.8	DEBUG_PROCESS Procedure	45-9
45.9	DEBUG_REGION Procedure Signature 1	45-9
45.10	DEBUG_REGION Procedure Signature 2	45-10
45.11	ESCAPE Function	45-11
45.12	EXECUTE_PLSQL_CODE Procedure (Deprecated)	45-11
45.13	GET_ATTRIBUTE_AS_NUMBER Function	45-12
45.14	GET_CURRENT_DATABASE_TYPE Function	45-13
45.15	GET_DATA Function Signature 1	45-13
45.16	GET_DATA Function Signature 2	45-15
45.17	GET_DATA2 Function Signature 1	45-17
45.18	GET_DATA2 Function Signature 2	45-20
45.19	GET_DISPLAY_DATA Function Signature 1	45-22
45.20	GET_DISPLAY_DATA Function Signature 2	45-23
45.21	GET_ELEMENT_ATTRIBUTES Function	45-25
45.22	GET_HTML_ATTR Function	45-27
45.23	GET_ORDERBY_NULLS_SUPPORT Function	45-27
45.24	GET_PLSQL_EXPR_RESULT_BOOLEAN Function	45-28
45.25	GET_PLSQL_EXPR_RESULT_CLOB Function	45-29
45.26	GET_PLSQL_EXPRESSION_RESULT Function	45-29
45.27	GET_PLSQL_FUNC_RESULT_BOOLEAN Function	45-30
45.28	GET_PLSQL_FUNC_RESULT_CLOB Function	45-31
45.29	GET_PLSQL_FUNCTION_RESULT Function	45-32
45.30	GET_POSITION_IN_LIST Function (Deprecated)	45-32
45.31	GET_SEARCH_STRING Function	45-33
45.32	GET_VALUE_AS_VARCHAR2 Function	45-34
45.33	GET_WEB_SOURCE_OPERATION Function	45-35
45.34	IS_EQUAL Function	45-36
45.35	IS_COMPONENT_USED Function	45-37
45.36	MAKE_REST_REQUEST Procedure Signature 1	45-38
45.37	MAKE_REST_REQUEST Procedure Signature 2	45-39
45.38	PAGE_ITEM_NAMES_TO_JQUERY Function (Deprecated)	45-41
45.39	PARSE_REFETCH_RESPONSE Function	45-42
45.40	PRINT_DISPLAY_ONLY Procedure Signature 1 (Deprecated)	45-44
45.41	PRINT_DISPLAY_ONLY Procedure Signature 2 (Deprecated)	45-45
45.42	PRINT_ESCAPED_VALUE Procedure Signature 1	45-46
45.43	PRINT_ESCAPED_VALUE Procedure Signature 2	45-46
45.44	PRINT_HIDDEN Procedure	45-47

45.45	PRINT_HIDDEN_IF_READONLY Procedure	45-47
45.46	PRINT_JSON_HTTP_HEADER Procedure	45-48
45.47	PRINT_LOV_AS_JSON Procedure	45-49
45.48	PRINT_OPTION Procedure	45-50
45.49	PRINT_READ_ONLY Procedure Signature 1	45-51
45.50	PRINT_READ_ONLY Procedure Signature 2	45-52
45.51	PROCESS_DML_RESPONSE Procedure	45-53
45.52	REPLACE_SUBSTITUTIONS Function	45-54
45.53	SET_COMPONENT_VALUES Procedure	45-55
45.54	SPLIT_MUTLIPLE_VALUE_TO_TABLE Function	45-57

46 APEX_PRINT

46.1	Constants	46-1
46.2	GENERATE_DOCUMENT Function Signature 1	46-2
46.3	GENERATE_DOCUMENT Function Signature 2	46-3
46.4	GENERATE_DOCUMENT Function Signature 3	46-4
46.5	GENERATE_DOCUMENT Function Signature 4	46-5
46.6	REMOVE_TEMPLATE Procedure	46-6
46.7	UPLOAD_TEMPLATE Function	46-6

47 APEX_PWA

47.1	GENERATE_PUSH_CREDENTIALS Procedure	47-1
47.2	HAS_PUSH_SUBSCRIPTION Function	47-1
47.3	PUSH_QUEUE Procedure	47-2
47.4	SEND_PUSH_NOTIFICATION Procedure	47-2
47.5	SUBSCRIBE_PUSH_NOTIFICATIONS Procedure	47-3
47.6	UNSUBSCRIBE_PUSH_NOTIFICATIONS Procedure	47-4

48 APEX_REGION

48.1	CLEAR Procedure	48-1
48.2	EXPORT_DATA Function	48-2
48.3	GET_ID Function Signature 1	48-4
48.4	GET_ID Function Signature 2	48-5
48.5	IS_READ_ONLY Function	48-6
48.6	OPEN_QUERY_CONTEXT Function	48-6
48.7	PURGE_CACHE Procedure	48-8
48.8	RESET Procedure	48-8

49 APEX_REST_SOURCE_SYNC

49.1	DISABLE Procedure	49-1
49.2	DYNAMIC_SYNCHRONIZE_DATA Procedure	49-2
49.3	ENABLE Procedure	49-3
49.4	GET_LAST_SYNC_TIMESTAMP Function	49-3
49.5	GET_SYNC_TABLE_DEFINITION_SQL Function	49-4
49.6	IS_RUNNING Function	49-5
49.7	RESCHEDULE Procedure	49-5
49.8	SYNCHRONIZE_DATA Procedure	49-6
49.9	SYNCHRONIZE_TABLE_DEFINITION Procedure	49-7

50 APEX_SEARCH

50.1	QUERY_EXPERT_SEARCH Function	50-1
50.2	QUERY_SEARCH_ENGINE Function	50-2
50.3	SEARCH Function	50-3

51 APEX_SESSION

51.1	ATTACH Procedure	51-1
51.2	CREATE_SESSION Procedure	51-2
51.3	DETACH Procedure	51-3
51.4	DELETE_SESSION Procedure	51-4
51.5	SET_DEBUG Procedure	51-5
51.6	SET_TENANT_ID Procedure	51-6
51.7	SET_TRACE Procedure	51-6

52 APEX_SESSION_STATE

52.1	Global Constants	52-1
52.2	Data Types	52-1
52.3	GET_CLOB Function	52-1
52.4	GET_NUMBER Function	52-2
52.5	GET_TIMESTAMP Function	52-2
52.6	GET_VALUE Function	52-2
52.7	GET_VARCHAR2 Function	52-3
52.8	SET_VALUE Procedure Signature 1	52-3
52.9	SET_VALUE Procedure Signature 2	52-3
52.10	SET_VALUE Procedure Signature 3	52-4

53 APEX_SHARED_COMPONENT

53.1	Global Constants	53-1
53.2	PUBLISH Procedure	53-2
53.3	REFRESH Procedure	53-2

54 APEX_SPATIAL

54.1	Data Types	54-1
54.2	CHANGE_GEOM_METADATA Procedure	54-2
54.3	CIRCLE_POLYGON Function	54-2
54.4	DELETE_GEOM_METADATA Procedure	54-3
54.5	INSERT_GEOM_METADATA Procedure	54-4
54.6	INSERT_GEOM_METADATA_LONLAT Procedure	54-5
54.7	POINT Function	54-6
54.8	RECTANGLE Function	54-6
54.9	SPATIAL_IS_AVAILABLE Function	54-7

55 APEX_STRING

55.1	FORMAT Function	55-2
55.2	GET_INITIALS Function	55-3
55.3	GET_SEARCHABLE_PHRASES Function	55-4
55.4	GREP Function Signature 1	55-5
55.5	GREP Function Signature 2	55-6
55.6	GREP Function Signature 3	55-7
55.7	INDEX_OF Function Signature 1	55-8
55.8	INDEX_OF Function Signature 2	55-8
55.9	JOIN_CLOB Function	55-9
55.10	JOIN_CLOBS Function	55-10
55.11	JOIN Function Signature 1	55-10
55.12	JOIN Function Signature 2	55-11
55.13	NEXT_CHUNK Function	55-11
55.14	PLIST_DELETE Procedure	55-12
55.15	PLIST_EXISTS Function	55-13
55.16	PLIST_GET Function	55-14
55.17	PLIST_GET_KEY Function	55-14
55.18	PLIST_PUSH Procedure	55-15
55.19	PLIST_PUT Function	55-16
55.20	PLIST_TO_JSON_CLOB Function	55-16
55.21	PUSH Procedure Signature 1	55-17
55.22	PUSH Procedure Signature 2	55-18

55.23	PUSH Procedure Signature 3	55-18
55.24	PUSH Procedure Signature 4	55-19
55.25	PUSH Procedure Signature 5	55-19
55.26	PUSH Procedure Signature 6	55-20
55.27	PUSH Procedure Signature 7	55-21
55.28	SHUFFLE Function	55-21
55.29	SHUFFLE Procedure	55-22
55.30	SPLIT Function Signature 1	55-22
55.31	SPLIT Function Signature 2	55-23
55.32	SPLIT_CLOBS Function	55-24
55.33	SPLIT_NUMBERS Function	55-24
55.34	STRING_TO_TABLE Function	55-25
55.35	TABLE_TO_CLOB Function	55-25
55.36	TABLE_TO_STRING Function	55-26

56 APEX_STRING_UTIL

56.1	DIFF Function	56-1
56.2	FIND_EMAIL_ADDRESSES Function	56-2
56.3	FIND_EMAIL_FROM Function	56-3
56.4	FIND_EMAIL_SUBJECT Function	56-4
56.5	FIND_IDENTIFIERS Function	56-4
56.6	FIND_LINKS Function	56-5
56.7	FIND_PHRASES Function	56-6
56.8	FIND_TAGS Function	56-7
56.9	GET_DOMAIN Function	56-8
56.10	GET_FILE_EXTENSION Function	56-8
56.11	GET_SLUG Function	56-9
56.12	PHRASE_EXISTS Function	56-9
56.13	REPLACE_WHITESPACE Function	56-10
56.14	TO_DISPLAY_FILESIZE Function	56-11

57 APEX_THEME

57.1	CLEAR_ALL_USERS_STYLE Procedure	57-1
57.2	CLEAR_USER_STYLE Procedure	57-2
57.3	DISABLE_USER_STYLE Procedure	57-2
57.4	ENABLE_USER_STYLE Procedure	57-3
57.5	GET_USER_STYLE Function	57-4
57.6	SET_CURRENT_STYLE Procedure	57-4
57.7	SET_SESSION_STYLE Procedure	57-5
57.8	SET_SESSION_STYLE_CSS Procedure	57-6

58 APEX_UI_DEFAULT_UPDATE

58.1	ADD_AD_COLUMN Procedure	58-2
58.2	ADD_AD_SYNONYM Procedure	58-3
58.3	DEL_AD_COLUMN Procedure	58-4
58.4	DEL_AD_SYNONYM Procedure	58-4
58.5	DEL_COLUMN Procedure	58-5
58.6	DEL_GROUP Procedure	58-5
58.7	DEL_TABLE Procedure	58-6
58.8	SYNCH_TABLE Procedure	58-7
58.9	UPD_AD_COLUMN Procedure	58-7
58.10	UPD_AD_SYNONYM Procedure	58-8
58.11	UPD_COLUMN Procedure	58-9
58.12	UPD_DISPLAY_IN_FORM Procedure	58-11
58.13	UPD_DISPLAY_IN_REPORT Procedure	58-11
58.14	UPD_FORM_REGION_TITLE Procedure	58-12
58.15	UPD_GROUP Procedure	58-13
58.16	UPD_ITEM_DISPLAY_HEIGHT Procedure	58-14
58.17	UPD_ITEM_DISPLAY_WIDTH Procedure	58-14
58.18	UPD_ITEM_FORMAT_MASK Procedure	58-15
58.19	UPD_ITEM_HELP Procedure	58-16
58.20	UPD_LABEL Procedure	58-16
58.21	UPD_REPORT_ALIGNMENT Procedure	58-17
58.22	UPD_REPORT_FORMAT_MASK Procedure	58-17
58.23	UPD_REPORT_REGION_TITLE Procedure	58-18
58.24	UPD_TABLE Procedure	58-19

59 APEX_UTIL

59.1	BLOB_TO_CLOB Function	59-5
59.2	CACHE_GET_DATE_OF_PAGE_CACHE Function	59-6
59.3	CACHE_GET_DATE_OF_REGION_CACHE Function	59-7
59.4	CACHE_PURGE_BY_APPLICATION Procedure	59-7
59.5	CACHE_PURGE_BY_PAGE Procedure	59-8
59.6	CACHE_PURGE_STALE Procedure	59-9
59.7	CHANGE_CURRENT_USER_PW Procedure	59-9
59.8	CHANGE_PASSWORD_ON_FIRST_USE Function	59-10
59.9	CLOB_TO_BLOB Function	59-11
59.10	CLOSE_OPEN_DB_LINKS Procedure	59-12
59.11	CLEAR_APP_CACHE Procedure	59-13

59.12	CLEAR_PAGE_CACHE Procedure	59-13
59.13	CLEAR_USER_CACHE Procedure	59-14
59.14	COUNT_CLICK Procedure	59-14
59.15	CREATE_USER Procedure	59-15
59.16	CREATE_USER_GROUP Procedure	59-19
59.17	CURRENT_USER_IN_GROUP Function	59-20
59.18	CUSTOM_CALENDAR Procedure (Deprecated)	59-21
59.19	DELETE_FEEDBACK Procedure	59-21
59.20	DELETE_FEEDBACK_ATTACHMENT Procedure	59-22
59.21	DELETE_USER_GROUP Procedure Signature 1	59-22
59.22	DELETE_USER_GROUP Procedure Signature 2	59-23
59.23	DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1	59-24
59.24	DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2	59-24
59.25	DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3	59-25
59.26	DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4	59-27
59.27	EDIT_USER Procedure	59-28
59.28	END_USER_ACCOUNT_DAYS_LEFT Function	59-32
59.29	EXPIRE_END_USER_ACCOUNT Procedure	59-33
59.30	EXPIRE_WORKSPACE_ACCOUNT Procedure	59-33
59.31	EXPORT_USERS Procedure	59-34
59.32	FEEDBACK_ENABLED Function	59-35
59.33	FETCH_APP_ITEM Function	59-35
59.34	FETCH_USER Procedure Signature 1	59-36
59.35	FETCH_USER Procedure Signature 2	59-39
59.36	FETCH_USER Procedure Signature 3	59-41
59.37	FIND_SECURITY_GROUP_ID Function	59-44
59.38	FIND_WORKSPACE Function	59-44
59.39	GET_ACCOUNT_LOCKED_STATUS Function	59-45
59.40	GET_APEX_OWNER Function	59-46
59.41	GET_APPLICATION_STATUS Function (Deprecated)	59-46
59.42	GET_ATTRIBUTE Function	59-47
59.43	GET_AUTHENTICATION_RESULT Function	59-48
59.44	GET_BLOB_FILE_SRC Function	59-48
59.45	GET_BUILD_OPTION_STATUS Function Signature 1 (Deprecated)	59-50
59.46	GET_BUILD_OPTION_STATUS Function Signature 2 (Deprecated)	59-51
59.47	GET_CURRENT_USER_ID Function	59-51
59.48	GET_DEFAULT_SCHEMA Function	59-52
59.49	GET_EDITION Function	59-53
59.50	GET_EMAIL Function	59-53
59.51	GET_FEEDBACK_FOLLOW_UP Function	59-54
59.52	GET_FILE Procedure	59-55
59.53	GET_FILE_ID Function	59-56

59.54	GET_FIRST_NAME Function	59-57
59.55	GET_GLOBAL_NOTIFICATION Function (Deprecated)	59-57
59.56	GET_GROUPS_USER_BELONGS_TO Function	59-58
59.57	GET_GROUP_ID Function	59-59
59.58	GET_GROUP_NAME Function	59-60
59.59	GET_HASH Function	59-60
59.60	GET_HIGH_CONTRAST_MODE_TOGGLE Function	59-61
59.61	GET_LAST_NAME Function	59-62
59.62	GET_NUMERIC_SESSION_STATE Function	59-63
59.63	GET_PREFERENCE Function	59-64
59.64	GET_PRINT_DOCUMENT Function Signature 1	59-65
59.65	GET_PRINT_DOCUMENT Function Signature 2	59-65
59.66	GET_PRINT_DOCUMENT Function Signature 3	59-66
59.67	GET_PRINT_DOCUMENT Function Signature 4	59-67
59.68	GET_SCREEN_READER_MODE_TOGGLE Function	59-68
59.69	GET_SESSION_LANG Function	59-68
59.70	GET_SESSION_STATE Function	59-69
59.71	GET_SESSION_TERRITORY Function	59-70
59.72	GET_SESSION_TIME_ZONE Function	59-70
59.73	GET_SINCE Function Signature 1	59-71
59.74	GET_SINCE Function Signature 2	59-72
59.75	GET_SUPPORTING_OBJECT_SCRIPT Function	59-72
59.76	GET_SUPPORTING_OBJECT_SCRIPT Procedure	59-73
59.77	GET_USER_ID Function	59-74
59.78	GET_USER_ROLES Function	59-75
59.79	GET_USERNAME Function	59-76
59.80	HOST_URL Function	59-77
59.81	HTML_PCT_GRAPH_MASK Function	59-78
59.82	INCREMENT_CALENDAR Procedure (Deprecated)	59-79
59.83	IR_CLEAR Procedure (Deprecated)	59-79
59.84	IR_DELETE_REPORT Procedure (Deprecated)	59-80
59.85	IR_DELETE_SUBSCRIPTION Procedure (Deprecated)	59-81
59.86	IR_FILTER Procedure (Deprecated)	59-81
59.87	IR_RESET Procedure (Deprecated)	59-83
59.88	IS_HIGH_CONTRAST_SESSION Function	59-84
59.89	IS_HIGH_CONTRAST_SESSION_YN Function	59-84
59.90	IS_LOGIN_PASSWORD_VALID Function	59-85
59.91	IS_SCREEN_READER_SESSION Function	59-86
59.92	IS_SCREEN_READER_SESSION_YN Function	59-86
59.93	IS_USERNAME_UNIQUE Function	59-87
59.94	KEYVAL_NUM Function	59-87
59.95	KEYVAL_VC2 Function	59-88

59.96	LOCK_ACCOUNT Procedure	59-89
59.97	PASSWORD_FIRST_USE_OCCURRED Function	59-89
59.98	PREPARE_URL Function	59-90
59.99	PRN Procedure	59-92
59.100	PUBLIC_CHECK_AUTHORIZATION Function (Deprecated)	59-93
59.101	PURGE_REGIONS_BY_APP Procedure	59-94
59.102	PURGE_REGIONS_BY_NAME Procedure	59-94
59.103	PURGE_REGIONS_BY_PAGE Procedure	59-95
59.104	REDIRECT_URL Procedure	59-96
59.105	REMOVE_PREFERENCE Procedure	59-96
59.106	REMOVE_SORT_PREFERENCES Procedure	59-97
59.107	REMOVE_USER Procedure Signature 1	59-97
59.108	REMOVE_USER Procedure Signature 2	59-98
59.109	REPLY_TO_FEEDBACK Procedure	59-99
59.110	RESET_AUTHORIZATIONS Procedure (Deprecated)	59-100
59.111	RESET_PASSWORD Procedure	59-100
59.112	RESET_PW Procedure	59-101
59.113	SAVEKEY_NUM Function	59-102
59.114	SAVEKEY_VC2 Function	59-103
59.115	SET_APP_BUILD_STATUS Procedure (Deprecated)	59-104
59.116	SET_APPLICATION_STATUS Procedure (Deprecated)	59-104
59.117	SET_ATTRIBUTE Procedure	59-106
59.118	SET_AUTHENTICATION_RESULT Procedure	59-107
59.119	SET_BUILD_OPTION_STATUS Procedure (Deprecated)	59-108
59.120	SET_CURRENT_THEME_STYLE Procedure (Deprecated)	59-109
59.121	SET_CUSTOM_AUTH_STATUS Procedure	59-111
59.122	SET_EDITION Procedure	59-112
59.123	SET_EMAIL Procedure	59-112
59.124	SET_FIRST_NAME Procedure	59-113
59.125	SET_GLOBAL_NOTIFICATION Procedure (Deprecated)	59-114
59.126	SET_GROUP_GROUP_GRANTS Procedure	59-115
59.127	SET_GROUP_USER_GRANTS Procedure	59-115
59.128	SET_LAST_NAME Procedure	59-116
59.129	SET_PARSING_SCHEMA_FOR_REQUEST Procedure	59-117
59.130	SET_PREFERENCE Procedure	59-118
59.131	SET_SECURITY_GROUP_ID Procedure	59-118
59.132	SET_SESSION_HIGH_CONTRAST_OFF Procedure	59-120
59.133	SET_SESSION_HIGH_CONTRAST_ON Procedure	59-120
59.134	SET_SESSION_LANG Procedure	59-120
59.135	SET_SESSION_LIFETIME_SECONDS Procedure	59-121
59.136	SET_SESSION_MAX_IDLE_SECONDS Procedure	59-122
59.137	SET_SESSION_SCREEN_READER_OFF Procedure	59-123

59.138	SET_SESSION_SCREEN_READER_ON Procedure	59-123
59.139	SET_SESSION_STATE Procedure	59-123
59.140	SET_SESSION_TERRITORY Procedure	59-124
59.141	SET_SESSION_TIME_ZONE Procedure	59-125
59.142	SET_USERNAME Procedure	59-125
59.143	SET_WORKSPACE Procedure	59-126
59.144	SHOW_HIGH_CONTRAST_MODE_TOGGLE Procedure	59-127
59.145	SHOW_SCREEN_READER_MODE_TOGGLE Procedure	59-128
59.146	STRING_TO_TABLE Function (Deprecated)	59-128
59.147	STRONG_PASSWORD_CHECK Procedure	59-129
59.148	STRONG_PASSWORD_VALIDATION Function	59-133
59.149	SUBMIT_FEEDBACK Procedure	59-134
59.150	SUBMIT_FEEDBACK_FOLLOWUP Procedure	59-135
59.151	TABLE_TO_STRING Function (Deprecated)	59-136
59.152	UNEXPIRE_END_USER_ACCOUNT Procedure	59-137
59.153	UNEXPIRE_WORKSPACE_ACCOUNT Procedure	59-138
59.154	UNLOCK_ACCOUNT Procedure	59-139
59.155	URL_ENCODE Function (Deprecated)	59-140
59.156	WORKSPACE_ACCOUNT_DAYS_LEFT Function	59-141

60 APEX_WEB_SERVICE

60.1	About the APEX_WEB_SERVICE API	60-2
60.1.1	Invoking a SOAP-style Web Service	60-2
60.1.2	Invoking a RESTful-style Web Service	60-4
60.1.3	Setting Cookies and HTTP Headers	60-5
60.1.4	Retrieving Cookies and HTTP Headers	60-5
60.2	About Web Credentials and APEX_WEB_SERVICE	60-6
60.3	Data Types	60-7
60.4	Global Variables	60-7
60.5	APPEND_TO_MULTIPART Procedure Signature 1	60-8
60.6	APPEND_TO_MULTIPART Procedure Signature 2	60-9
60.7	BLOB2CLOBBASE64 Function	60-9
60.8	CLEAR_REQUEST_COOKIES Procedure	60-10
60.9	CLEAR_REQUEST_HEADERS Procedure	60-11
60.10	CLOBBASE642BLOB Function	60-11
60.11	GENERATE_REQUEST_BODY Function	60-12
60.12	GET_REQUEST_HEADER Function	60-12
60.13	MAKE_REQUEST Function Signature 1	60-13
60.14	MAKE_REQUEST Function Signature 2	60-15
60.15	MAKE_REQUEST Procedure Signature 1	60-16
60.16	MAKE_REQUEST Procedure Signature 2	60-18

60.17	MAKE_REST_REQUEST Function	60-19
60.18	MAKE_REST_REQUEST_B Function	60-20
60.19	OAUTH_AUTHENTICATE_CREDENTIAL Procedure	60-22
60.20	OAUTH_AUTHENTICATE Procedure Signature 1	60-23
60.21	OAUTH_AUTHENTICATE Procedure Signature 2 (Deprecated)	60-24
60.22	OAUTH_GET_LAST_TOKEN Function	60-25
60.23	OAUTH_SET_TOKEN Procedure	60-25
60.24	PARSE_RESPONSE Function	60-26
60.25	PARSE_RESPONSE_CLOB Function	60-26
60.26	PARSE_XML Function	60-27
60.27	PARSE_XML_CLOB Function	60-28
60.28	REMOVE_REQUEST_HEADER Procedure	60-29
60.29	SET_REQUEST_ECID_CONTEXT Procedure	60-30
60.30	SET_REQUEST_HEADERS Procedure	60-31

61 APEX_WORKFLOW

61.1	Constants and Data Types	61-1
61.2	CONTINUE_ACTIVITY Procedure Signature 1	61-3
61.3	CONTINUE_ACTIVITY Procedure Signature 2	61-4
61.4	GET_LOV_ACTIVITY_STATE Function	61-5
61.5	GET_LOV_WORKFLOW_STATE Function	61-6
61.6	GET_NEXT_PURGE_TIMESTAMP Function	61-6
61.7	GET_VARIABLE_CLOB_VALUE Function	61-7
61.8	GET_VARIABLE_VALUE Function	61-7
61.9	GET_WORKFLOW_STATE Function	61-8
61.10	GET_WORKFLOWS Function	61-9
61.11	IS_ADMIN Function	61-10
61.12	IS_ALLOWED Function	61-11
61.13	IS_OF_PARTICIPANT_TYPE Function	61-12
61.14	REFRESH_PARTICIPANTS Procedure	61-12
61.15	REMOVE_DEVELOPMENT_INSTANCES Procedure	61-13
61.16	RESUME Procedure	61-13
61.17	RETRY Procedure	61-14
61.18	SET_LOG_LEVEL Procedure	61-15
61.19	START_WORKFLOW Function	61-15
61.20	SUSPEND Procedure	61-16
61.21	TERMINATE Procedure	61-17
61.22	TERMINATEFAULTEDWORKFLOWS Procedure	61-17
61.23	UPDATE_VARIABLES Procedure	61-18

62 APEX_ZIP

62.1	Data Types	62-1
62.2	ADD_FILE Procedure	62-2
62.3	FINISH Procedure	62-3
62.4	GET_DIR_ENTRIES Function	62-3
62.5	GET_FILE_CONTENT Function Signature 1 (Deprecated)	62-5
62.6	GET_FILE_CONTENT Function Signature 2	62-5
62.7	GET_FILES Function (Deprecated)	62-6

63 JavaScript APIs

Index

Preface

Oracle APEX API Reference describes the available Application Programming Interfaces (APIs) when programming in the Oracle APEX environment. To utilize these APIs, such as APEX_JSON, when not developing with APEX, you must install APEX into the database.

- [Audience](#)
- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
- [Related Documents](#)
- [Conventions](#)

Audience

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

See Also:

Oracle APEX App Builder User's Guide

Documentation Accessibility

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

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners,

we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For more information, see these Oracle resources:

- *Oracle APEX Release Notes*
- *Oracle APEX Installation Guide*
- *Oracle APEX App Builder User's Guide*
- *Oracle APEX Administration Guide*
- *Oracle APEX SQL Workshop Guide*
- *Oracle APEX End User's Guide*
- *Oracle Database Concepts*
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Database PL/SQL Language Reference*

Conventions

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

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

The following text conventions are used in this document:

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

Changes in Release 24.2 for Oracle APEX API Reference

All content in *Oracle APEX API Reference* has been updated to reflect release 24.2 functionality.

New Features and Updates

The following topics have been added or updated for this release:

- APEX_AI (Updates)
 - CHAT Function Signature 1 (Updated) - Parameter `p_prompt` is now CLOB to accept larger texts.
 - CHAT Function Signature 2 (New) - Chats with a generative AI service given a prompt and potential earlier responses.
 - GENERATE Function Signature 1 (Updated) - New parameter `p_system_prompt`. Parameter `p_prompt` is now CLOB to accept larger texts.
 - GENERATE Function Signature 2 (New) - Generates a response for a given prompt.
 - GET_VECTOR_EMBEDDINGS Function Signatures 1, 2, and 3 (New) - Receives the embedding from a vector provider for a given term for Vector Search.
- APEX_APPLICATION_ADMIN (Updates)
 - SET_REMOTE_SERVER Procedure (New) - Sets the base URL, HTTPS host, and other attributes for remote servers.
- APEX_CREDENTIAL (Updates)
 - SET_SCOPE Procedure (New) - Changes the "scope" attribute of a Web Credential.
- APEX_EXEC (Updates)
 - Global Constants (Updated) - New DML Location constants `REGION_SOURCE`, `DUALITY_VIEW`, and `JSON_COLLECTION`. New Column and Filter constants for Vector Search: `c_filter_vector_type`, `c_data_type_array`, and `c_data_type_vector`, and new category of Vector Search subtypes.
 - ADD_FILTER Procedure (Updated) - New signature and parameters for Vector Search.
 - DESCRIBE_QUERY Function Signature 2 (Updated) - New parameters `p_duality_view_static_id` and `p_json_source_static_id`.
 - OPEN_DUALITY_VIEW_DML_CONTEXT Function (New) - Opens a DML context based on a Duality View source.
 - OPEN_JSON_SOURCE_DML_CONTEXT Function (New) - Opens a DML context based on a JSON source.
 - OPEN_QUERY_CONTEXT Function Signature 1 (Updated) - New parameters `p_duality_view_static_id` and `p_json_source_static_id`.

- OPEN_QUERY_CONTEXT Function Signature 2 (Updated) - New parameter `p_control_break`.
- OPEN_REST_SOURCE_QUERY Function (Updated) - New parameter `p_control_break`.
- PURGE_JSON_SOURCE_CACHE Procedure (New) - Purge the local cache for a Duality View of JSON Sources based on REST Enabled SQL.
- PURGE_DUALITY_VIEW_CACHE Procedure (New) - Purge the local cache for a Duality View of JSON Sources based on REST Enabled SQL.
- APEX_EXPORT (Updates)
 - GET_APPLICATION Function (Updated) - New parameter `p_with_runtime_instances`.
- APEX_HUMAN_TASK (Updates)
 - GET_LOV_TYPE Function (New) - Gets the list of value data for the task attribute type.
 - SET_INITIATOR_CAN_COMPLETE Procedure (New) - Updates the `initiator_can_complete` attribute of a task.
- APEX_INSTANCE_ADMIN (Updates)
 - Available Parameters (Updated) - Parameter `ALLOW_HOSTING_EXTENSIONS` can now be configured to host extensions and publish links to other workspaces.
- APEX_IR (Updates)
 - Constants and Data Types (New) - List of constants and data types used by the APEX_IR package.
- APEX_LANG (Updates)
 - CREATE_MESSAGE Procedure (Updated) - New parameters `p_comment` and `p_metadata`.
 - GET_MESSAGE Function (New) - Translates text strings (or messages) generated from PL/SQL-stored procedures, functions, triggers, packaged procedures, and functions.
 - UPDATE_MESSAGE Procedure Signature 2 (New) - Updates a translatable text message and its attributes for the specified application.
- APEX_PAGE (Updates)
 - GET_URL Function (Updated) - New parameter `p_absolute_url`.
- APEX_PLUGIN (Updates)
 - About Configuring Flexible Remote Servers in APEX (New) - Flexible remote servers use a configuration procedure to change the URL endpoint of the server.
 - Global Constants (Updated) - New subtype `t_escape_mode`.
 - Data Types (Updated) - New types `t_remote_server_info` and `t_remote_server_config`.
 - `t_plugin_attributes` (Updated) - New `get_varchar2` parameters `p_default_value`, `p_do_substitutions`, `p_do_serveronly_substitutions`, and `p_substitutions_escape_mode`. New functions `get_number` and `get_boolean`.
 - `t_item` (Updated) - New member attributes.
- APEX_REGION (Updates)

- GET_ID Function Signature 1 (New) - Gets the region ID based on the dom static ID.
- GET_ID Function Signature 2 (New) - Gets the region ID based on the region name.
- APEX_SHARED_COMPONENT (New) - APIs to work with shared components. Only available in full APEX installations.
 - Global Constants (New)
 - REFRESH Procedure (New)
 - PUBLISH Procedure (New)
- APEX_STRING (Updates)
 - PLIST_EXISTS Function (New) - Returns whether a key exists in the property list.
 - PLIST_GET_KEY Function (New) - Gets the first property list key that maps to a given value.
- APEX_UTIL (Updates)
 - DELETE_FEEDBACK Procedure (New) - Deletes feedback with the specified ID.
 - DELETE_FEEDBACK_ATTACHMENT Procedure (New) - Deletes the attachment of a feedback with the specified ID.
 - GET_APEX_OWNER Function (New) - Returns the name of the schema containing the APEX engine.
 - REPLY_TO_FEEDBACK Procedure (New) - Submits a reply to a feedback.
- APEX_WORKFLOW (Updates)
 - GET_VARIABLE_CLOB_VALUE Function (New) - Gets the CLOB value of a workflow variable.
 - RESUME Procedure (Updated) - New parameter `p_activity_static_id`.

Deprecated and Desupported Features

The following APIs are deprecated as of this release:

- APEX_LANG
 - MESSAGE Function (Deprecated) - Use GET_MESSAGE instead.
- APEX_PLUGIN_UTIL
 - EXECUTE_PLSQL_CODE Procedure (Deprecated) - Use APEX_EXEC.EXECUTE_PLSQL Signature 1 or 2 instead.
- APEX_UTIL
 - CUSTOM_CALENDAR Procedure (Deprecated)
 - INCREMENT_CALENDAR Procedure (Deprecated)

See Deprecated Features and Desupported Features in *Oracle APEX Release Notes*.

APEX_ACL

The `APEX_ACL` package provides utilities that you can use when programming in the Oracle APEX environment related to the Shared Components for application access control. You can use the `APEX_ACL` package to add, remove, or replace user roles. You can also use the `INSTEAD OF` trigger on the `APEX_APPL_ACL_USERS` view to edit user roles with DML statements (`INSERT`, `UPDATE`, and `DELETE`).

If the package is used outside of an 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.

Use the related APEX views `APEX_APPL_ACL_ROLES`, `APEX_APPL_ACL_USERS`, and `APEX_APPL_USER_ROLES` to get more information on application users and roles.

- [ADD_USER_ROLE Procedure Signature 1](#)
- [ADD_USER_ROLE Procedure Signature 2](#)
- [HAS_USER_ANY_ROLES Function](#)
- [HAS_USER_ROLE Function](#)
- [IS_ROLE_REMOVED_FROM_USER 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 apex_application.g_flow_id,
    p_user_name        IN VARCHAR2,
    p_role_id          IN NUMBER )
```

Parameters

Parameter	Description
<code>p_application_id</code>	The application ID for which you want to assign a role to a user. Defaults to the current application.
<code>p_user_name</code>	The case insensitive name of the application user to assign the role to.
<code>p_role_id</code>	The ID of the role.

Example

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

```
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

```
APEX_ACL.ADD_USER_ROLE (
    p_application_id IN NUMBER  DEFAULT apex_application.g_flow_id,
    p_user_name       IN VARCHAR2,
    p_role_static_id IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to assign a role to a user. Defaults to the current application.
p_user_name	The case-insensitive name of the application user to assign the role to.
p_role_static_id	The case-insensitive name of the role static ID.

Example

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

```
BEGIN
    APEX_ACL.ADD_USER_ROLE (
        p_application_id => 255,
        p_user_name      => 'SCOTT',
        p_role_static_id => 'ADMINISTRATOR' );
END;
```

2.3 HAS_USER_ANY_ROLES Function

This function returns TRUE when the specified user is assigned to any application role. This function can be used to check if a user is permitted to access an application.

Syntax

```
APEX_ACL.HAS_USER_ANY_ROLES (
    p_application_id IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_user_name       IN VARCHAR2 DEFAULT apex_application.g_user )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to check if a user is assigned to any application role. Defaults to the current application.
p_user_name	The case insensitive name of the application user to check. Defaults to the current logged-in user.

Example

The following example uses the HAS_USER_ANY_ROLES function to check if the user name SCOTT is assigned to any application role in application 255.

```
DECLARE
    l_has_user_any_roles boolean := false;
BEGIN
    l_has_user_any_roles := APEX_ACL.HAS_USER_ANY_ROLES (
        p_application_id => 255,
        p_user_name       => 'SCOTT' );

    IF NOT l_has_user_any_roles THEN
        raise_application_error(-20001, 'Scott is not assigned to any
application role' );
    END IF;
END;
```

2.4 HAS_USER_ROLE Function

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

Syntax

```
APEX_ACL.HAS_USER_ROLE (
    p_application_id IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_user_name       IN VARCHAR2 DEFAULT apex_application.g_user,
    p_role_static_id IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to check if a user is assigned to the specific role. Defaults to the current application.

Parameter	Description
p_user_name	The case insensitive name of the application user to check. It defaults to the current logged in user.
p_role_static_id	The case insensitive name of the role static ID.

Example

The following example uses the HAS_USER_ROLE function to check if the user name 'SCOTT' is assigned to any 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 IS_ROLE_REMOVED_FROM_USER Function

This function checks if a role is removed from a user. This function returns TRUE if a specific role is removed from the list of new role IDs for the user.

This function is used to ensure that a user cannot remove a role identified by p_role_static_id from him/herself.

Syntax

```

APEX_ACL.IS_ROLE_REMOVED_FROM_USER (
    p_application_id    IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_user_name         IN VARCHAR2,
    p_role_static_id   IN VARCHAR2,
    p_role_ids          IN apex_t_number )
RETURN BOOLEAN;

```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to check if a specific role removed from the list of roles was from a user. It defaults to the current application.
p_user_name	The case insensitive name of the application user to check.
p_role_static_id	The case insensitive name of the role static ID to check if it is removed.
p_role_ids	The array of NUMBER type new role IDs the user is assigned to.

Returns

Returns TRUE when p_user_name currently has the role identified by p_role_static_id but the roles identified by p_role_ids do not include the role identified by p_role_static_id.

Return FALSE in all other cases.

Example

The following example uses the IS_ROLE_REMOVED_FROM_USER function to ensure the current user of the app who has the ADMINISTRATOR role does not remove him/herself from the role when updating or deleting the access to the app.

```
BEGIN
    IF :P1_USER_NAME = :APP_USER
        and apex_acl.is_role_removed_from_user (
            p_application_id => :APP_ID,
            p_user_name      => :APP_USER,
            p_role_static_id => 'ADMINISTRATOR',
            p_role_ids       => apex_string.split_numbers(
                p_str => case when :REQUEST =
'DELETE' THEN
                            null
                        ELSE
                            :P1_ROLE_IDS
                        END,
                p_sep => ':') ) THEN
        raise_application_error(-20001, 'You cannot remove administrator role
from yourself.' );
    END IF;
END;
```

2.6 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 apex_application.g_flow_id,
    p_user_name      IN VARCHAR2,
    p_role_id        IN NUMBER );
```

Parameters

Parameter	Description
p_application_id	The application ID from which you want to remove an assigned role from a user. Defaults to the current application.
p_user_name	The case insensitive name of the application user to remove the role from.
p_role_id	The ID of the role.

Example

The following example uses the `REMOVE_USER_ROLE` procedure to remove the role ID of 2505704029884282 from the user name 'SCOTT' in application 255.

```
BEGIN
    APEX_ACL.REMOVE_USER_ROLE (
        p_application_id => 255,
        p_user_name      => 'SCOTT',
        p_role_id        => 2505704029884282 );
END;
```

2.7 REMOVE_USER_ROLE Procedure Signature 2

This procedure removes an assigned role from a user.

Syntax

```
APEX_ACL.REMOVE_USER_ROLE (
    p_application_id IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_user_name       IN VARCHAR2,
    p_role_static_id IN VARCHAR2 )
```

Parameters

Parameter	Description
<code>p_application_id</code>	The application ID from which you want to remove an assigned role from a user. It defaults to the current application.
<code>p_user_name</code>	The case insensitive name of the application user to remove the role from.
<code>p_role_static_id</code>	The case insensitive name of the role static ID.

Example

The following example uses the `REMOVE_USER_ROLE` procedure to remove the 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;
```

2.8 REPLACE_USER_ROLES Procedure Signature 1

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

Syntax

```
APEX_ACL.REPLACE_USER_ROLES (
    p_application_id IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_user_name      IN VARCHAR2,
    p_role_ids       IN apex_t_number );
```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to replace the user roles. Defaults to the current application.
p_user_name	The case insensitive name of the application user to replace the role.
p_role_ids	The array of NUMBER type role IDs.

Example

The following example uses the REPLACE_USER_ROLES procedure to replace existing roles with new role IDs of 2505704029884282 and 345029884282 for the user name 'SCOTT' in application 255.

```
BEGIN
    APEX_ACL.REPLACE_USER_ROLES (
        p_application_id => 255,
        p_user_name      => 'SCOTT',
        p_role_ids       => apex_t_number( 2505704029884282, 345029884282 ) );
END;
```

2.9 REPLACE_USER_ROLES Procedure Signature 2

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

Syntax

```
APEX_ACL.REPLACE_USER_ROLES (
    p_application_id IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_user_name      IN VARCHAR2,
    p_role_static_ids IN apex_t_varchar2 );
```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to replace the user roles. Defaults to the current application.
p_user_name	The case insensitive name of the application user to replace the role.
p_role_static_ids	The array of case-insensitive VARCHAR2-type role static IDs.

Example

The following example uses the `REPLACE_USER_ROLES` procedure to replace existing roles with new role static IDs of 'ADMINISTRATOR' and 'CONTRIBUTOR' for the user name 'SCOTT' in application 255.

```
BEGIN
    APEX_ACL.REPLACE_USER_ROLES (
        p_application_id => 255,
        p_user_name      => 'SCOTT',
        p_role_static_ids => apex_t_varchar2( 'ADMINISTRATOR',
        'CONTRIBUTOR' ) );
END;
```

2.10 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 apex_application.g_flow_id,
    p_user_name       IN VARCHAR2 );
```

Parameters

Parameter	Description
p_application_id	The application ID for which you want to remove all assigned roles from a user. Defaults to the current application.
p_user_name	The case-insensitive name of the application user to remove all assigned roles from.

Example

The following example uses the `REMOVE_ALL_USER_ROLES` procedure to remove all assigned roles from the user name 'SCOTT' in application 255.

```
BEGIN
    APEX_ACL.REMOVE_ALL_USER_ROLES (
        p_application_id => 255,
        p_user_name      => 'SCOTT' );
END;
```

3

APEX_AI

APEX_AI contains the APIs for Oracle APEX Generative AI.

- Constants
- Data Types
- CHAT Function Signature 1
- CHAT Function Signature 2
- GENERATE Function Signature 1
- GENERATE Function Signature 2
- GET_VECTOR_EMBEDDINGS Function Signature 1
- GET_VECTOR_EMBEDDINGS Function Signature 2
- GET_VECTOR_EMBEDDINGS Function Signature 3
- IS_ENABLED Function
- IS_USER_CONSENT_NEEDED Function
- REVOKE_USER_CONSENT Procedure
- REVOKE_USER_CONSENT_FOR_ALL Procedure
- SET_USER_CONSENT Procedure

3.1 Constants

The APEX_AI package uses the following constants.

```
c_chat_messages      t_chat_messages;
```

3.2 Data Types

The APEX_AI package uses the following data types.

```
subtype t_chat_role is varchar2(30);

type t_chat_message is record (
    chat_role    t_chat_role,
    message      clob );

type t_chat_messages is table of t_chat_message index by pls_integer;
```

3.3 CHAT Function Signature 1

This function chats with a Generative AI service given a prompt and potential earlier responses.

Syntax

```
APEX_AI.CHAT (
    p_prompt          IN          CLOB,
    p_system_prompt   IN          VARCHAR2      DEFAULT NULL,
    p_service_static_id IN          VARCHAR2      DEFAULT NULL,
    p_temperature     IN          NUMBER        DEFAULT NULL,
    p_messages        IN OUT NOCOPY t_chat_messages )
RETURN CLOB;
```

Parameters

Parameter	Description
p_prompt	The user prompt.
p_system_prompt	(Optional) System prompt to pass. Some Generative AI services (such as OpenAI) support the use of passing a system prompt to set the context of a conversation.
p_service_static_id	The Generative AI Service static ID. If not provided, uses the app's default AI Service.
p_temperature	The temperature to use. How the temperature is interpreted depends on the Generative AI Service implementation. Higher temperatures result in more "creative" responses. See the documentation of the Generative AI provider for details and allowed values.
p_messages	(Optional) The responses from an earlier conversation. Responses of procedure chat and nl2sql are automatically added to p_responses.

Returns

The response for the given prompt and type.

Example

The following example chats with the configured Generative AI Service MY_AI_SERVICE. In the first interaction, a system prompt is given and then in further interactions the context is passed to the Generative AI service in the form of parameter p_messages.

```
DECLARE
    l_messages apex_ai.t_chat_messages;
    l_response1 clob;
    l_response2 clob;
BEGIN
    l_response1 := apex_ai.chat(
        p_prompt          => 'What is Oracle APEX',
        p_system_prompt   => 'I am an expert in Low Code Application Platforms',
        p_service_static_id => 'MY_AI_SERVICE',
        p_messages         => l_messages);
    l_response2 := apex_ai.chat(
        p_prompt          => 'What is new in 23.2',
        p_service_static_id => 'MY_AI_SERVICE',
        p_messages         => l_messages);
END;
```

3.4 CHAT Function Signature 2

This function chats with a Generative AI service given a prompt and potential earlier responses.

Syntax

```
APEX_AI.CHAT (
    p_config_static_id  IN          VARCHAR2,
    p_prompt            IN          CLOB,
    p_messages          IN OUT NOCOPY t_chat_messages )
    RETURN CLOB;
```

Parameters

Parameter	Description
p_config_static_id	The static ID of the AI configuration defined under the application's Shared Components.
p_prompt	The user prompt.
p_messages	(Optional) The responses from an earlier conversation. Responses are automatically added to p_responses for an easy conversational experience.

Returns

The response for the given prompt and type.

Example

The following example chats with the assistant configured as my-oracle-assistant where in the first interaction a system prompt is given and then in further interactions the context is passed to the generative AI service in the form of parameter p_messages.

```
DECLARE
    l_messages  t_chat_messages := c_chat_messages;
    l_response1 clob;
    l_response2 clob;
BEGIN
    l_response1 := apex_ai.chat(
        p_config_static_id => 'my-oracle-assistant',
        p_prompt           => 'What is Oracle APEX',
        p_messages         => l_messages);
    l_response2 := apex_ai.chat(
        p_config_static_id => 'my-oracle-assistant',
        p_prompt           => 'What is new in 23.2',
        p_messages         => l_messages)
END;
```

3.5 GENERATE Function Signature 1

This function generates a response for a given prompt.

Syntax

```
APEX_AI.GENERATE (
    p_prompt          IN CLOB,
    p_system_prompt   IN VARCHAR2      DEFAULT NULL,
    p_service_static_id IN VARCHAR2      DEFAULT NULL,
    p_temperature     IN NUMBER        DEFAULT NULL )
RETURN CLOB;
```

Parameters

Parameter	Description
p_prompt	The user prompt.
p_system_prompt	(Optional) System prompt to pass. Some Generative AI services (such as OpenAI) support the use of passing a system prompt to set the context of a request.
p_service_static_id	The Generative AI Service static ID. If not provided, uses the app's default AI Service.
p_temperature	The temperature to use. How the temperature is interpreted depends on the Generative AI Service implementation. Higher temperatures result in more "creative" responses. See the documentation of the Generative AI provider for details and allowed values.

Returns

The response for the given prompt and type.

Example

The following example generates a response with the configured Generative AI Service MY_AI_SERVICE for the given prompt.

```
DECLARE
    l_response clob;
BEGIN
    l_response := apex_ai.generate(
        p_prompt          => 'What is Oracle APEX',
        p_service_static_id => 'MY_AI_SERVICE');
END;
```

3.6 GENERATE Function Signature 2

This function generates a response for a given prompt.

Syntax

```
APEX_AI.GENERATE (
    p_config_static_id  IN          VARCHAR2 )
    p_prompt            IN          VARCHAR2 )
RETURN CLOB;
```

Parameters

Parameter	Description
p_config_static_id	The static ID of the AI configuration defined under the application's Shared Components.
p_prompt	The user prompt.

Example

The following example generates a response using the AI service configured via AI configuration with static ID low_code_expert.

```
DECLARE
    l_response clob;
BEGIN
    l_response := apex_ai.generate(
        p_config_static_id => 'low_code_expert',
        p_prompt           => 'What is Oracle APEX' );
END;
```

3.7 GET_VECTOR_EMBEDDINGS Function Signature 1

This function receives the embedding from a vector provider for a given term.

Syntax

```
APEX_AI.GET_VECTOR_EMBEDDINGS (
    p_value          IN CLOB,
    p_service_static_id IN VARCHAR2 )
RETURN VECTOR;
```

Parameters

Parameter	Description
p_value	The textual value for which the embedding is to be determined.
p_service_static_id	The Vector Provider static ID.

Returns

The embedding for the given value.

Example

The following example uses the Vector Provider "MY_ONNX_VECTOR_PROVIDER" to receive an embedding.

```
DECLARE
    l_vector vector;
BEGIN
    l_vector := apex_ai.get_vector_embeddings(
        p_value           => 'What is Oracle APEX',
```

```

    p_service_static_id => 'MY_ONNX_VECTOR_PROVIDER' );
END;

```

3.8 GET_VECTOR_EMBEDDINGS Function Signature 2

This function receives the embedding from a vector provider for a given term.

Syntax

```

APEX_AI.GET_VECTOR_EMBEDDINGS (
    p_value          IN CLOB,
    p_local_llm_owner IN VARCHAR2,
    p_local_llm_name   IN VARCHAR2 )
RETURN VECTOR;

```

Parameters

Parameter	Description
p_value	The textual value for which the embedding is to be determined.
p_local_llm_owner	The owner name of the local ONNX model to be used to receive the embeddings.
p_local_llm_name	The name of the local ONNX model to be used to receive the embeddings.

Returns

The embedding for the given value.

3.9 GET_VECTOR_EMBEDDINGS Function Signature 3

This function receives the embedding from a vector provider for a given term.

Syntax

```

APEX_AI.GET_VECTOR_EMBEDDINGS (
    p_value          IN CLOB,
    p_function_name  IN VARCHAR2 )
RETURN VECTOR;

```

Parameters

Parameter	Description
p_value	The textual value for which the embedding is to be determined.
p_function_name	The name of a custom PL/SQL function which converts an end user input to an embedding. The specified function needs to take the end user input (p_value) as VARCHAR2 and returns a VECTOR type as a result.

Returns

The embedding for the given value.

3.10 IS_ENABLED Function

This function returns whether Generative AI features are enabled for the current Oracle APEX Workspace.

Syntax

```
APEX_AI.IS_ENABLED
RETURN BOOLEAN;
```

Parameters

None.

Returns

TRUE if Generative AI features are enabled for the current workspace. Otherwise, FALSE.

Example

```
DECLARE
    l_is_ai_enabled boolean;
BEGIN
    l_is_ai_enabled := apex_ai.is_enabled;
    dbms_output.put_line('AI is enabled: ' || case l_is_ai_enabled when true
then 'Yes' else 'No' end);
END;
```

3.11 IS_USER_CONSENT_NEEDED Function

This function returns whether a consent screen is shown to the user before interacting with the AI.

Syntax

```
APEX_AI.IS_USER_CONSENT_NEEDED (
    p_user_name      IN  VARCHAR2      DEFAULT {the current user},
    p_application_id IN  NUMBER        DEFAULT {the current application} )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_user_name	The user name. Defaults to the current user.
p_application_id	The application ID. Defaults to the current application.

Returns

TRUE if an AI consent message exists and if the user has **not** already consented. Otherwise, FALSE.

Example

The following example checks whether user consent is needed for the current user and application.

```
DECLARE
    l_user_consent_needed boolean;
BEGIN
    l_user_consent_needed := apex_ai.is_user_consent_needed;
END;
```

3.12 REVOKE_USER_CONSENT Procedure

This procedure removes the AI user preference storing the usage consent.

Syntax

```
APEX_AI.REVOKE_USER_CONSENT (
    p_user_name      IN VARCHAR2,
    p_application_id IN NUMBER )
```

Parameters

Parameter	Description
p_user_name	The username.
p_application_id	The application ID.

Example

```
BEGIN
    apex_ai.revoke_user_consent(
        p_user_name      => 'STIGER',
        p_application_id => 100);
END;
```

3.13 REVOKE_USER_CONSENT_FOR_ALL Procedure

This procedure removes the AI user preference storing the usage consent for all users.

Syntax

```
APEX_AI.REVOKE_USER_CONSENT_FOR_ALL (
    p_application_id IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

```
BEGIN
    apex_ai.revoke_user_consent_for_all(
        p_application_id => 100);
END;
```

3.14 SET_USER_CONSENT Procedure

This procedure marks the user as having consented to the use of AI.

If done once either by the user via the UI or via this API, the user is no longer prompted to consent when interacting with AI.

Syntax

```
APEX_AI.SET_USER_CONSENT (
    p_user_name      IN  VARCHAR2,
    p_application_id IN  NUMBER )
```

Parameters

Parameter	Description
p_user_name	The user name.
p_application_id	The application ID.

Example

```
BEGIN
    apex_ai.set_user_consent(
        p_user_name      => 'STIGER',
        p_application_id => 100);
END;
```

4

APEX_APP_OBJECT_DEPENDENCY

This API enables building the Application Database Object Dependencies report on demand.

It scans the application (or a page in the application) for all database objects that it depends on (including but not limited to tables, views, procedures, functions, packages, and synonyms), whether these dependencies are in forms, reports, PL/SQL regions, conditions, plugins, or elsewhere. It can also be used to detect invalid SQL and PL/SQL in the application.

The result of the scan may be viewed by querying the following views:

- APEX_USED_DB_OBJECT_COMP_PROPS - all application SQL and PL/SQL found
- APEX_USED_DB_OBJECTS - all database objects referred to
- APEX_USED_DB_OBJ_DEPENDENCIES - all dependencies found

In the event that a fragment of SQL or PL/SQL is invalid (such as a required object is missing), some dependencies may not be detected. The compilation error message may be queried in the APEX_USED_DB_OBJECT_COMP_PROPS view.

- Constants
- [CLEAR_CACHE Procedure](#)
- [SCAN Procedure](#)

4.1 Constants

The APEX_APP_OBJECT_DEPENDENCY package uses the following constants.

```
c_option_all          constant varchar2(30) := 'ALL';
c_option_dependencies constant varchar2(30) := 'DEPENDENCIES';
c_option_identifiers constant varchar2(30) := 'IDENTIFIERS';
c_option_errors       constant varchar2(30) := 'ERRORS';
```

4.2 CLEAR_CACHE Procedure

This procedure removes all cached dependency report data for a given application.

Syntax

```
APEX_APP_OBJECT_DEPENDENCY.CLEAR_CACHE (
    p_application_id    IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	ID of the application for which cache is cleared.

Example

```
BEGIN
    apex_app_object_dependency.clear_cache ( p_application_id => :app_id );
END;
```

4.3 SCAN Procedure

This procedure generates the object dependency report.

It scans the application or a page in the application for all database objects that it depends on (including tables, views, procedures, functions, packages, and synonyms) whether these dependencies are in forms, reports, PL/SQL regions, conditions, plugins, or elsewhere.

The results are visible by querying the following views:

- APEX_USED_DB_OBJECT_COMP_PROPS - all application SQL and PL/SQL found
- APEX_USED_DB_OBJECTS - all database objects referred to
- APEX_USED_DB_OBJ_DEPENDENCIES - all dependencies found

In the event that a fragment of SQL or PL/SQL is invalid (for example, a required object is missing), the dependencies are not detected. The compilation error message may be queried in the APEX_USED_DB_OBJECT_COMP_PROPS view.

The results of the scan are saved until:

- a new scan initiates
- apex_app_object_dependency.clear_cache is called
- the Oracle APEX instance is upgraded

PL/Scope

The scanner only detects dependencies specific to functions and procedures within packages compiled with PL/Scope. Before starting the scan, you may choose to compile the schema(s) of interest with PL/Scope:

```
alter session set plsScope_settings='identifiers:all';
exec sys.dbms_utility.compile_schema(user, true);
alter session set plsScope_settings='identifiers:none';
```

This may take some time to run depending on the size of the codebase.

For packages not compiled with PL/Scope, the scanner only detects a dependency on the package; but the report does not list each method referenced within it.

Requirements

The application owner schema requires CREATE PROCEDURE privilege.

Known Limitations

- Does not detect dependencies within SQL generated dynamically (such as using execute immediate or dbms_sql).

- Does not detect dependencies arising from Function Returning SQL or Function Body Returning SQL (the functions are not executed so the code that they generate is not scanned for dependencies).
- Does not detect recursive dependencies (other database objects referred to by the objects detected in the scan). Tip: Recursive dependencies may be found by querying the `USER_DEPENDENCIES` database view.
- Does not detect dependencies in Supporting Object scripts, including those that may arise in Required Object Names, Install Scripts, Upgrade Scripts, or Deinstall Scripts.

Syntax

```
APEX_APP_OBJECT_DEPENDENCY.SCAN (
    p_application_id IN NUMBER,
    p_page_id         IN NUMBER    DEFAULT NULL,
    p_options         IN VARCHAR2 DEFAULT c_option_all )
```

Parameters

Parameter	Description
p_application_id	ID of the application to be analyzed.
p_page_id	Set this parameter to analyze a single page of an application.
p_options	Options include: <ul style="list-style-type: none">• <code>c_option_all</code> - (Default) Scan all sources.• <code>c_option_dependencies</code> - Only scan for top-level dependencies with <code>dba_dependencies</code>.• <code>c_option_identifiers</code> - Scan for detailed dependencies with PL/Scope where available. This enables detection of dependencies on columns in tables and views, and also member functions and procedures within packages compiled with <code>identifiers:all</code>.• <code>c_option_errors</code> - Scan neither (report compilation errors only).

Example

```
BEGIN
    apex_app_object_dependency.scan ( p_application_id => :app_id );
END;
```

5

APEX_APP_SETTING

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

- [GET_VALUE Function](#)
- [SET_VALUE Procedure](#)

5.1 GET_VALUE Function

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

Syntax

```
APEX_APP_SETTING.GET_VALUE (
    p_name          IN VARCHAR2
    p_raise_error   IN BOOLEAN DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameters	Description
p_name	The case insensitive name of the application setting. An error raises if: <ul style="list-style-type: none">• the application setting name does not exist• the build option associated with the application setting is disabled
p_raise_error	If TRUE, the procedure raises an error if an application setting with a passed name does not exist.

Example

The following example uses 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;
```

5.2 SET_VALUE Procedure

This procedure changes the application setting value in the current application. If the setting is subscribed from another app, this API will not update the setting value. If the setting is subscribed and `p_raise_error` is set to TRUE, this API raises an error.

Syntax

```
APEX_APP_SETTING.SET_VALUE (
    p_name      IN VARCHAR2,
    p_value     IN VARCHAR2,
    p_raise_error IN BOOLEAN DEFAULT FALSE )
```

Parameters

Parameters	Description
p_name	The case-insensitive name of the application setting. An error raises if: <ul style="list-style-type: none">• the application setting name does not exist• the build option associated with the application setting is disabled
p_value	The value of the application setting. An error raises if: <ul style="list-style-type: none">• the value is set to required, but a null value passes• the valid values are defined, but the value is not in one of the valid values
p_raise_error	If set to TRUE and an error occurs, then this procedure raises an error message. If set to FALSE, all error messages are suppressed. In either case, this API never updates application setting values when an error occurs.

Example

The following example uses the SET_VALUE procedure to set the value of the application setting "ACCESS_CONTROL_ENABLED."

```
BEGIN
    APEX_APP_SETTING.SET_VALUE (
        p_name  => 'ACCESS_CONTROL_ENABLED',
        p_value => 'Y' );
END;
```

6

APEX_APPLICATION

The `APEX_APPLICATION` package is a PL/SQL package that implements the Oracle APEX rendering engine. You can use this package to take advantage of many global variables.

- [Working with G_Fnn Arrays \(Legacy\)](#)
- [Global Variables](#)
- [HELP Procedure](#)
- [STOP_APEX_ENGINE Procedure](#)

6.1 Working with G_Fnn Arrays (Legacy)

Important:

Support for `G_Fnn` arrays is legacy and will be removed in a future release. Oracle recommends using interactive grids instead.

The `APEX_APPLICATION.G_Fnn` arrays (where *nn* ranges from 01 to 50) are used with `APEX_ITEM` functions to enable the dynamic generation of HTML form elements to an APEX page (such as `APEX_ITEM.TEXT` and `APEX_ITEM.SELECT_LIST`). On Page Submit, the item values are sent to the server and provided as the `APEX_APPLICATION.G_Fnn` arrays.

Only use `APEX_APPLICATION.G_Fnn` in an `APEX_ITEM` context. For other contexts (such as plain array processing for PL/SQL code) use the `APEX_T_VARCHAR2` type and the procedures and functions within the `APEX_STRING` package.

Note:

When working with `APEX_APPLICATION.G_Fnn`, the `TABLE_TO_STRING` and `STRING_TO_TABLE` functions in `APEX_UTIL` are deprecated. Use `APEX_STRING.TABLE_TO_STRING` and `APEX_STRING.STRING_TO_TABLE` instead.

Referencing G_Fnn Arrays

The following example uses `APEX_ITEM` to manually create a tabular form on the `EMP` table. Note that the `ename`, `sal`, and `comm` columns use the `APEX_ITEM.TEXT` function to generate an HTML text field for each row. Note also that each item in the query is passed a unique `p_idx` parameter to ensure that each column is stored in its own array.

1. On a new page, add a classic report with a SQL Query such as the following example:

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno) ||
```

```
APEX_ITEM.TEXT(2,ename) ename,
APEX_ITEM.TEXT(3,job) job,
mgr,
APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
APEX_ITEM.TEXT(5,sal) sal,
APEX_ITEM.TEXT(6,comm) comm,
deptno
FROM emp
ORDER BY 1
```

2. Disable "Escape Special Characters" for all report columns (under the Security property in Page Designer).
3. Add a Submit button to the page.
4. Run the application.

Referencing Values Within an On Submit Process

You can reference the values posted by the tabular 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, the following code block collects all employee names as a text block and stores it as the value of the `P3_G_F01_CONTENTS` item:

```
:P3_G_F01_CONTENTS := '';
for i IN 1..APEX_APPLICATION.G_F01.COUNT LOOP
    :P3_G_F01_CONTENTS := :P3_G_F01_CONTENTS
        || 'element'||i||' has a value of '||
APEX_APPLICATION.G_F02(i) || chr(10);
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.

See Also:

- [APEX_IG](#)
- [APEX_ITEM \(Legacy\)](#)
- [APEX_STRING](#)
- [STRING_TO_TABLE Function](#)
- [TABLE_TO_STRING Function](#)

6.2 Global Variables

Global Variable	Description
G_USER	Specifies the currently logged in user.
G_FLOW_ID	Specifies the ID of the currently running application.
G_FLOW_STEP_ID	Specifies the ID of the currently running page.
G_FLOW_OWNER	Defaults to the application's parsing schema. Use #OWNER# to reference this value in SQL queries and PL/SQL.

 **Note:**

Changing G_FLOW_OWNER at runtime does not change the parsing schema.

G_REQUEST	Specifies the value of the request variable most recently passed to or set within the show or accept modules.
G_BROWSER_LANGUAGE	Refers to the web browser's current language preference.
G_DEBUG	Refers to whether debugging is switched on or off. Valid values for the DEBUG flag are Yes or No. Enabling debug shows details about application processing.
G_HOME_LINK	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 Oracle APEX engine redirects to this location.
G_LOGIN_URL	Used to display a link to a login page for users that are not currently logged in.
G_IMAGE_PREFIX	Refers to the virtual path the web server uses to point to the images directory distributed with APEX.
G_FLOW_SCHEMA_OWNER	Refers to the owner of the APEX schema.
G_PRINTER_FRIENDLY	Refers to whether the APEX engine is running in print view mode. This setting can be referenced in conditions to eliminate elements not desired in a printed document from a page.
G_PROXY_SERVER	Refers to the application attribute Proxy Server.
G_SYSDATE	Refers to the current date on the database server. G_SYSDATE uses the DATE datatype.
G_PUBLIC_USER	Refers to the Oracle schema used to connect to the database through the database access descriptor (DAD).
G_GLOBAL_NOTIFICATIONS	Specifies the application's global notification attribute.
G_X01, ... G_X10	Specifies the values of the X01, ... X10 variables most recently passed to or set within the show or accept modules. You typically use these variables in On-Demand AJAX processes.

6.3 HELP Procedure

This procedure 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

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

Parameter	Description
p_before_item_html	Use this parameter to include HTML before every item help text for item-level help. This parameter is ignored if p_show_item_help is set to NO.
p_after_item_html	Use this parameter to include HTML after every item help text for item-level help. This parameter is ignored if p_show_item_help is set to NO.

Example

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

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

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

```
APEX_APPLICATION.HELP(
    p_flow_id => :APP_ID,
    p_flow_step_id => :REQUEST,
    p_before_region_html => '<p><br/><table class="u-info"
width="100%"><tr><td><b>',
    p_after_prompt_html  => '</b></p>&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. Add a "Navigation Bar" link to this page, ensuring that the REQUEST value set in the link is &APP_PAGE_ID.

6.4 STOP_APEX_ENGINE Procedure

This procedure signals the Oracle APEX 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 tells the browser to redirect to `http://apex.oracle.com/` and immediately stops further processing. 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 APEX engine exception
    WHEN others THEN
        ...; -- code to handle the exception
END;
```

APEX_APPLICATION_ADMIN

The APEX_APPLICATION_ADMIN package provides APIs to modify application attributes of installed Oracle APEX applications.

- Constants and Data Types
- GET_APPLICATION_ALIAS Function
- GET_APPLICATION_NAME Function
- GET_APPLICATION_STATUS Function
- GET_APPLICATION_VERSION Function
- GET_AUTHENTICATION_SCHEME Function
- GET_BUILD_OPTION_STATUS Function Signature 1
- GET_BUILD_OPTION_STATUS Function Signature 2
- GET_BUILD_STATUS Function
- GET_GLOBAL_NOTIFICATION Function
- GET_FILE_STORAGE Function
- GET_IMAGE_PREFIX Function
- GET_MAX_SCHEDULER_JOBS Function
- GET_NO_PROXY_DOMAINS Function
- GET_PARSING_SCHEMA Function
- GET_PASS_ECID Function
- GET_PROXY_SERVER Function
- SET_APPLICATION_ALIAS Procedure
- SET_APPLICATION_NAME Procedure
- SET_APPLICATION_STATUS Procedure
- SET_APPLICATION_VERSION Procedure
- SET_AUTHENTICATION_SCHEME Procedure
- SET_BUILD_OPTION_STATUS Procedure
- SET_BUILD_STATUS Procedure
- SET_FILE_STORAGE Procedure
- SET_GLOBAL_NOTIFICATION Procedure
- SET_IMAGE_PREFIX Procedure
- SET_MAX_SCHEDULER_JOBS Procedure
- SET_PARSING_SCHEMA Procedure
- SET_PASS_ECID Procedure
- SET_PROXY_SERVER Procedure

- `SET_REMOTE_SERVER` Procedure

7.1 Constants and Data Types

The `APEX_APPLICATION_ADMIN` package uses the following constants and data types.

Application Status

```
subtype t_app_status is varchar2(30);
c_app_available           constant t_app_status := 'AVAILABLE';
c_app_available_with_edit_link constant t_app_status :=
'AVAILABLE_W_EDIT_LINK';
c_app_available_devs_only   constant t_app_status := 'DEVELOPERS_ONLY';
c_app_restricted_access    constant t_app_status := 'RESTRICTED_ACCESS';
c_app_unavailable          constant t_app_status := 'UNAVAILABLE';
c_app_unavailable_redirect constant t_app_status := 'UNAVAILABLE_URL';
c_app_unavailable_show_plsql constant t_app_status := 'UNAVAILABLE_PLSQL';
```

Build Status

```
subtype t_build_status is varchar2(30);
c_build_status_run_and_build  constant t_build_status := 'RUN_AND_BUILD';
c_build_status_run_only       constant t_build_status := 'RUN_ONLY';
```

Build Option Status

```
subtype t_build_option_status is varchar2(30);
c_build_option_status_include constant t_build_option_status := 'INCLUDE';
c_build_option_status_exclude  constant t_build_option_status := 'EXCLUDE';
```

Storage Type

```
subtype t_storage_type is varchar2(30);
c_file_storage_oci      constant t_storage_type := 'OBJECT_STORE';
c_file_storage_db        constant t_storage_type := 'DB';
```

7.2 GET_APPLICATION_ALIAS Function

This function retrieves the application alias.

Syntax

```
APEX_APPLICATION_ADMIN.GET_APPLICATION_ALIAS (
  p_application_id    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
<code>p_application_id</code>	The application ID.

Example

The following example returns the value of the application alias.

```
DECLARE
    l_application_alias varchar2(255);
BEGIN
    l_application_alias := apex_application_admin.get_application_alias (
        p_application_id => 100 );
END;
```

 **See Also:**

[SET_APPLICATION_ALIAS Procedure](#)

7.3 GET_APPLICATION_NAME Function

This function retrieves the application name.

Syntax

```
APEX_APPLICATION_ADMIN.GET_APPLICATION_NAME (
    p_application_id      IN NUMBER )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example returns the value of the application name.

```
DECLARE
    l_application_name varchar2(255);
BEGIN
    l_application_name := apex_application_admin.get_application_name (
        p_application_id => 100 );
END;
```

 **See Also:**

[SET_APPLICATION_NAME Procedure](#)

7.4 GET_APPLICATION_STATUS Function

This function retrieves the application_status (such as Available, Unavailable). Returns t_app_status.

Syntax

```
APEX_APPLICATION_ADMIN.GET_APPLICATION_STATUS (
    p_application_id    IN NUMBER )
    RETURN t_app_status;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example gets the application status for application 100 and works with one of the constants to act on the result.

```
DECLARE
    l_app_status apex_application_admin.t_app_status;
BEGIN
    apex_util.set_workspace('YOUR_WORKSPACE_NAME');
    l_app_status := apex_application_admin.get_application_status (
        p_application_id => 100 );
    IF l_app_status = apex_application_admin.c_app_available_with_edit_link THEN
        dbms_output.put_line(l_app_status);
        -- your custom code here...
    END IF;
END;
```

See Also:

[SET_APPLICATION_STATUS Procedure](#)

7.5 GET_APPLICATION_VERSION Function

This function retrieves the version of an application.

Syntax

```
APEX_APPLICATION_ADMIN.GET_APPLICATION_VERSION (
    p_application_id    IN NUMBER )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example prints the application version.

```
select apex_application_admin.get_application_version(100) from sys.dual
```



[SET_APPLICATION_VERSION Procedure](#)

7.6 GET_AUTHENTICATION_SCHEME Function

This function retrieves the authentication scheme of an application.

Syntax

```
APEX_APPLICATION_ADMIN.GET_AUTHENTICATION_SCHEME (
    p_application_id    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example prints the authentication scheme override.

```
select apex_application_admin.get_authentication_scheme(100) from sys.dual
```



[SET_AUTHENTICATION_SCHEME Procedure](#)

7.7 GET_BUILD_OPTION_STATUS Function Signature 1

This function retrieves the status of a build option by ID.

Syntax

```
APEX_APPLICATION_ADMIN.GET_BUILD_OPTION_STATUS (
    p_application_id    IN NUMBER,
    p_id                IN NUMBER )
RETURN t_build_option_status;
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_id	The build option ID.

 **See Also:**

[SET_BUILD_OPTION_STATUS Procedure](#)

7.8 GET_BUILD_OPTION_STATUS Function Signature 2

This function retrieves the status of a build option by name.

Syntax

```
APEX_APPLICATION_ADMIN.GET_BUILD_OPTION_STATUS (
    p_application_id    IN NUMBER,
    p_build_option_name IN VARCHAR2 )
RETURN t_build_option_status;
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_build_option_name	The build option name.

Returns

INCLUDE - The build option is "Include" (associated components are enabled and part of the application).

EXCLUDE - The build option is "Exclude" (associated components are disabled and not part of the application).

 **See Also:**

[SET_BUILD_OPTION_STATUS Procedure](#)

7.9 GET_BUILD_STATUS Function

This function retrieves the application build status.

Syntax

```
APEX_APPLICATION_ADMIN.GET_BUILD_STATUS (
    p_application_id    IN NUMBER )
RETURN t_build_status;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example returns the value of the application build status.

```
DECLARE
    l_application_build_status apex_application_admin.t_build_status;
BEGIN
    l_application_build_status := apex_application_admin.get_build_status (
                                p_application_id => 100 );
END;
```

See Also:

[SET_BUILD_STATUS Procedure](#)

7.10 GET_GLOBAL_NOTIFICATION Function

This function retrieves the global notification message. This is the message displayed in page #GLOBALNOTIFICATION# substitution string.

Syntax

```
APEX_APPLICATION_ADMIN.GET_GLOBAL_NOTIFICATION (
    p_application_id    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

 See Also:

[SET_GLOBAL_NOTIFICATION Procedure](#)

7.11 GET_FILE_STORAGE Function

This function retrieves the static ID of the file storage remote server of an application.

If database file storage is used, returns NULL.

Syntax

```
APEX_APPLICATION_ADMIN.GET_FILE_STORAGE (
    p_application_id    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The ID of the application.

Returns

The static ID of the file storage remote server, if OCI file storage is used. NULL otherwise.

Example

The following example prints the static ID of application file storage remote server setting.

```
select apex_application_admin.get_file_storage(100) from sys.dual
```

 See Also:

- [SET_FILE_STORAGE Procedure](#)

7.12 GET_IMAGE_PREFIX Function

This function retrieves the image prefix.

Syntax

```
APEX_APPLICATION_ADMIN.GET_IMAGE_PREFIX (
    p_application_id    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example returns the value of the image prefix.

```
DECLARE
    l_image_prefix varchar2(255);
BEGIN
    l_image_prefix := apex_application_admin.get_image_prefix (
                      p_application_id => 100 );
END;
```

 **See Also:**

[SET_IMAGE_PREFIX Procedure](#)

7.13 GET_MAX_SCHEDULER_JOBS Function

This function fetches the application attribute "Maximum Scheduler Jobs."

This function also indicates how many scheduler jobs can run at the same time to execute background page processes.

Syntax

```
APEX_APPLICATION_ADMIN.GET_MAX_SCHEDULER_JOBS (
    p_application_id      IN NUMBER )
RETURN NUMBER
```

Parameters

Parameter	Description
p_application_id	The application ID.

 **See Also:**

- [SET_MAX_SCHEDULER_JOBS Procedure](#)

7.14 GET_NO_PROXY_DOMAINS Function

This function retrieves the no proxy domains attribute of an application.

Syntax

```
APEX_APPLICATION_ADMIN.GET_NO_PROXY_DOMAINS (
    p_application_id      IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Returns

This function returns a comma-delimited list of domains for which the proxy server cannot be used. The no proxy domains attribute cannot be more than 500 characters.

See Also:

- [GET_PROXY_SERVER Function](#)
- [SET_PROXY_SERVER Procedure](#)

7.15 GET_PARSING_SCHEMA Function

This function retrieves the parsing schema (or "owner") of an application.

Syntax

```
APEX_APPLICATION_ADMIN.GET_PARSING_SCHEMA (
    p_application_id      IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

The following example returns the value of the application schema for application 100.

```
DECLARE
    l_schema varchar2(30);
```

```
BEGIN
    l_schema := apex_application_admin.get_parsing_schema(
        p_application_id => 100 );
END;
```

 **See Also:**

[SET_PARSING_SCHEMA Procedure](#)

7.16 GET_PASS_ECID Function

This function retrieves the application security attribute "Pass ECID" (Execution Context ID). This indicates whether to pass the ECID to the external web services for end-to-end tracing.

Syntax

```
APEX_APPLICATION_ADMIN.GET_PASS_ECID (
    p_application_id    IN NUMBER )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_application_id	The application ID.

 **See Also:**

[SET_PASS_ECID Procedure](#)

7.17 GET_PROXY_SERVER Function

This function retrieves the proxy server attribute of an application.

Syntax

```
APEX_APPLICATION_ADMIN.GET_PROXY_SERVER (
    p_application_id    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

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

```
DECLARE
    l_proxy_server varchar2(255);
BEGIN
    l_proxy_server := apex_application_admin.get_proxy_server (
        p_application_id => 100 );
END;
```

 **See Also:**

- [GET_NO_PROXY_DOMAINS Function](#)
- [SET_PROXY_SERVER Procedure](#)

7.18 SET_APPLICATION_ALIAS Procedure

This procedure sets the application alias.

Syntax

```
APEX_APPLICATION_ADMIN.SET_APPLICATION_ALIAS (
    p_application_id    IN NUMBER,
    p_application_alias IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_application_alias	The application alias. Cannot be more than 255 characters. Cannot be null.

Example

The following example sets the application alias to "EXECUTIVE-DASHBOARD" for application 100.

```
DECLARE
    c_id      constant number      := 100;
    c_alias   constant varchar2(255) := 'EXECUTIVE-DASHBOARD';
BEGIN
    apex_application_admin.set_application_alias (
        p_application_id    => c_id,
        p_application_alias => c_alias );
END;
```

 See Also:[GET_APPLICATION_ALIAS Function](#)

7.19 SET_APPLICATION_NAME Procedure

This procedure sets the application name.

Syntax

```
APEX_APPLICATION_ADMIN.SET_APPLICATION_NAME (
    p_application_id    IN NUMBER,
    p_application_name  IN VARCHAR2 );
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_application_name	The application name. Cannot be longer than 255 characters. Cannot be null.

Example

The following example sets the application name to "Executive Dashboard" for application 100.

```
DECLARE
    c_id    constant number      := 100;
    c_name  constant varchar2(255) := 'Executive Dashboard';
BEGIN
    apex_application_admin.set_application_name (
        p_application_id  => c_id,
        p_application_name => c_name );
END;
```

 See Also:[GET_APPLICATION_NAME Function](#)

7.20 SET_APPLICATION_STATUS Procedure

This procedure sets the status of the application.

Syntax

```
APEX_APPLICATION_ADMIN.SET_APPLICATION_STATUS (
    p_application_id    IN NUMBER,
```

```

    p_application_status      IN t_app_status,
    --
    p_allowed_users_list    IN apex_t_varchar2 DEFAULT NULL,
    --
    p_message                IN VARCHAR2 DEFAULT NULL,
    p_plsql_code              IN VARCHAR2 DEFAULT NULL,
    p_redirect_url            IN VARCHAR2 DEFAULT NULL )

```

Parameters

Parameter	Description
p_application_id	The application ID.
p_application_status	New status to set application to. Values include: <ul style="list-style-type: none"> apex_application_admin.c_app_available - Application is available with no restrictions. apex_application_admin.c_app_available_with_edit_link - Application is available with no restrictions. Developer Toolbar displays for developers. apex_application_admin.c_app_available_devs_only - Application only available to developers. apex_application_admin.c_app_restricted_access - Application only available to users in p_allowed_users_list. apex_application_admin.c_app_unavailable - Application unavailable. Message shown in p_message. apex_application_admin.c_app_unavailable_redirect - Application unavailable. Redirected to URL provided in p_redirect_url. apex_application_admin.c_app_unavailable_show_plsql - Application unavailable. Message shown from PL/SQL block in p_plsql_code.
p_allowed_users_list	An apex_t_varchar2 list of users which are allowed to access the application when p_application_status = c_app_restricted_access.
p_message	Message shown to users when p_application_status = c_app_unavailable.
p_plsql_code	Message shown to users when p_application_status = c_app_unavailable_show_plsql.
p_redirect_url	URL to redirect to when p_application_status = c_app_unavailable_redirect.

Example

The following example sets the status for application 100 to "restricted access" and permits only USER1 and USER2 to use it.

```

BEGIN
    apex_util.set_workspace('YOUR_WORKSPACE_NAME');
    apex_application_admin.set_application_status (
        p_application_id      => 100,
        p_application_status  => apex_application_admin.c_app_restricted_access,
        p_allowed_users_list => apex_t_varchar2('USER1','USER2') );
    COMMIT;
END;

```

 See Also:[GET_APPLICATION_STATUS Function](#)

7.21 SET_APPLICATION_VERSION Procedure

This procedure sets the version of an application.

Syntax

```
APEX_APPLICATION_ADMIN.SET_APPLICATION_VERSION (
    p_application_id IN NUMBER,
    p_version        IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_version	The version information. Cannot be longer than 255 characters.

Example

The following example sets the version for application 100.

```
BEGIN
    apex_application_admin.set_application_version (
        p_application_id => 100,
        p_version         => 'Release 1.0' );
END;
```

 See Also:[GET_APPLICATION_VERSION Function](#)

7.22 SET_AUTHENTICATION_SCHEME Procedure

This procedure sets the authentication scheme of an application.

Syntax

```
APEX_APPLICATION_ADMIN.SET_AUTHENTICATION_SCHEME (
    p_application_id IN NUMBER,
    p_name           IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_name	The name of the authentication scheme to be activated.
	This new authentication scheme must exist in the application. If null, the active authentication scheme remains unchanged.

Example

The following example activates authentication scheme "SSO-Production" for application 100.

```
BEGIN
    apex_application_admin.set_authentication_scheme (
        p_application_id => 100,
        p_name             => 'SSO-Production' );
END;
```

 **See Also:**

[GET_AUTHENTICATION_SCHEME Function](#)

7.23 SET_BUILD_OPTION_STATUS Procedure

This procedure sets the status of a build option.

Syntax

```
APEX_APPLICATION_ADMIN.SET_BUILD_OPTION_STATUS (
    p_application_id    IN NUMBER,
    p_id                IN NUMBER,
    p_build_status      IN t_build_option_status )
```

Parameters

Parameter	Description
p_app	The application ID.
p_id	The build option ID.
p_build_status	Status with possible values: <ul style="list-style-type: none"> • apex_application_admin.c_build_option_status_include • apex_application_admin.c_build_option_status_exclude

 See Also:

- [GET_BUILD_OPTION_STATUS Function Signature 1](#)
- [GET_BUILD_OPTION_STATUS Function Signature 2](#)

7.24 SET_BUILD_STATUS Procedure

This procedure sets the application build status.

Syntax

```
APEX_APPLICATION_ADMIN.SET_BUILD_STATUS (
    p_application_id IN NUMBER,
    p_build_status    IN t_build_status )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_build_status	New build status to set application to. Values include: <ul style="list-style-type: none"> • RUN_AND_BUILD - Developers and users can both run and develop the application. • RUN_ONLY - Users can only run the application. Developers cannot edit the application.

Example

The following example sets build status for app 100 to "RUN_ONLY."

```
BEGIN
    apex_application_admin.set_build_status (
        p_application_id => 100,
        p_build_status    => 'RUN_ONLY' );
END;
/
```

 See Also:

- [GET_BUILD_STATUS Function](#)

7.25 SET_FILE_STORAGE Procedure

This procedure sets the file storage type to use either the local database or OCI Object store. If Object store is chosen, you must pass the static ID of the remote server pointing to the object store bucket.

Syntax

```
APEX_APPLICATION_ADMIN.SET_FILE_STORAGE (
    p_application_id          IN NUMBER,
    p_storage_type             IN t_storage_type,
    p_remote_server_static_id IN VARCHAR2 DEFAULT NULL,
    p_migrate_files            IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_application_id	The ID of the application.
p_storage_type	Whether to use database or OCI storage for files.
p_remote_server_static_id	If OCI is used, Static ID of the remote server.
p_migrate_files	If TRUE, migrates application files from the application export to specified file storage server.

Example

The following example sets the file storage to OCI during import, uses the remote server with the static ID `bucket-app-files-myapp`, and uploads all files contained in the export file.

```
BEGIN
    apex_application_admin.set_file_storage(
        p_application_id      => 100,
        p_storage_type         =>
    apex_application_admin.c_file_storage_oci,
        p_remote_server_static_id => 'bucket-app-files-myapp',
        p_migrate_files           => true );
END;
```

See Also:

- [GET_FILE_STORAGE Function](#)

7.26 SET_GLOBAL_NOTIFICATION Procedure

This procedure sets the global notification message. This is the message displayed in page `#GLOBALNOTIFICATION#` substitution string.

Syntax

```
APEX_APPLICATION_ADMIN.SET_GLOBAL_NOTIFICATION (
    p_application_id          IN NUMBER,
    p_global_notification_message IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_global_notification_message	The new global notification message.

 **See Also:**[GET_GLOBAL_NOTIFICATION Function](#)

7.27 SET_IMAGE_PREFIX Procedure

This procedure sets the application image prefix.

Syntax

```
APEX_APPLICATION_ADMIN.SET_IMAGE_PREFIX (
    p_application_id IN NUMBER,
    p_image_prefix    IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_image_prefix	The image prefix. Cannot be longer than 255 characters.

Example

The following example sets the application image prefix to "/static/" for application 100.

```
DECLARE
    c_id          constant number      := 100;
    c_image_prefix constant varchar2(255) := '/static/';
BEGIN
    apex_application_admin.set_image_prefix (
        p_application_id => c_id,
        p_image_prefix   => c_image_prefix );
END;
```

 **See Also:**[GET_IMAGE_PREFIX Function](#)

7.28 SET_MAX_SCHEDULER_JOBS Procedure

This procedure sets the application attribute "Maximum Scheduler Jobs."

This procedure also indicates how many scheduler jobs can run at the same time to execute background page processes.

Syntax

```
APEX_APPLICATION_ADMIN.SET_MAX_SCHEDULER_JOBS (
    p_application_id      IN NUMBER,
    p_max_scheduler_jobs  IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_max_scheduler_jobs	Maximum number of scheduler jobs running for this application at the same time.

Example

The following example sets the maximum scheduler jobs for app 100 to 5.

```
BEGIN
    apex_application_admin.set_max_scheduler_jobs(100, 5);
END;
```

See Also:

- [GET_MAX_SCHEDULER_JOBS Function](#)

7.29 SET_PARSING_SCHEMA Procedure

This procedure sets the parsing schema ("owner") of an application.

The database user of the schema must already exist and the schema name must already be mapped to the workspace.

Syntax

```
APEX_APPLICATION_ADMIN.SET_PARSING_SCHEMA (
    p_application_id IN NUMBER,
    p_schema         IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_schema	The schema name.

Example

The following example sets the parsing schema to "EXAMPLE" for application 100.

```
BEGIN
    apex_application_admin.set_parsing_schema (
        p_application_id => 100,
        p_schema          => 'EXAMPLE' );
END;
```

 **See Also:**

[GET_PARSING_SCHEMA Function](#)

7.30 SET_PASS_ECID Procedure

This procedure sets the application Security attribute "Pass ECID" (Execution Context ID). Indicates whether to pass the ECID to the external web services for end-to-end tracing.

Syntax

```
APEX_APPLICATION_ADMIN.SET_PASS_ECID (
    p_application_id IN NUMBER,
    p_pass_ecid      IN BOOLEAN )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_pass_ecid	Boolean value: TRUE or FALSE.

Example

```
BEGIN
    apex_application_admin.set_pass_ecid(100, true);
END;
```

 See Also:

[GET_PASS_ECID Function](#)

7.31 SET_PROXY_SERVER Procedure

This procedure sets the proxy server attributes of an application.

Syntax

```
APEX_APPLICATION_ADMIN.SET_PROXY_SERVER (
    p_application_id    IN NUMBER,
    p_proxy_server      IN VARCHAR2 ,
    p_no_proxy_domains IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_proxy_server	The proxy server. There is no default value. The proxy server must be fewer than 255 characters and must exclude any protocol prefix such as http://. The following example is valid: www-proxy.example.com
p_no_proxy_domains	Comma-delimited list of domains for which the proxy server is invalid. Default value is null. Cannot be more than 500 characters.

Example

The following example sets the value of the proxy for an application.

```
BEGIN
    apex_application_admin.set_proxy_server (
        p_proxy_server => 'www-proxy.example.com' );
END;
```

 See Also:

- [GET_NO_PROXY_DOMAINS Function](#)
- [GET_PROXY_SERVER Function](#)

7.32 SET_REMOTE_SERVER Procedure

This procedure sets the base URL, HTTPS host, and other attributes for remote servers. Remote servers are identified by their Static ID.

Syntax

```
APEX_APPLICATION_ADMIN.SET_REMOTE_SERVER (
    p_static_id      IN VARCHAR2,
    p_base_url       IN VARCHAR2,
    p_https_host     IN VARCHAR2 DEFAULT NULL,
    --
    p_default_database IN VARCHAR2 DEFAULT NULL,
    p_mysql_sql_modes IN VARCHAR2 DEFAULT NULL,
    --
    p_ords_timezone   IN VARCHAR2 DEFAULT NULL,
    --
    p_ai_model_name   IN VARCHAR2 DEFAULT NULL,
    p_ai_http_headers IN CLOB      DEFAULT NULL,
    p_ai_attributes    IN CLOB      DEFAULT NULL )
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object.
p_base_url	New base URL to use for this remote server object.
p_https_host	New HTTPS host property to use for this remote server object.
p_default_database	Default database to use when connecting. Currently only supported for MySQL databases.
p_mysql_sql_modes	SQL modes to use when connecting to a MySQL database.
p_ords_timezone	Time zone in which the remote ORDS server runs. Only relevant for REST-Enabled SQL services, when remote ORDS does not run in UTC time zone.
p_ai_model_name	The AI model to use when requesting a response from a generative AI service.
p_ai_http_headers	HTTP headers to use when making a request to a generative AI service.
p_ai_attributes	Attributes in JSON format to use when making a request to a generative AI service.

Example

```
BEGIN
    apex_application_admin.set_remote_server(
        p_static_id => 'MY_REMOTE_SERVER',
        p_base_url => 'http://production.example.com' );
END;
```

APEX_APPLICATION_INSTALL

The APEX_APPLICATION_INSTALL package provides many methods to modify application attributes during the Oracle APEX application installation process.

- [About the APEX_APPLICATION_INSTALL API](#)
- [Attributes Manipulated by APEX_APPLICATION_INSTALL](#)
- Import Data Types
- 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_AUTHENTICATION_SCHEME Function
- GET_AUTO_INSTALL_SUP_OBJ Function
- GET_BUILD_STATUS Function
- GET_IMAGE_PREFIX Function
- GET_INFO Function
- GET_KEEP_BACKGROUND_EXECS Function
- GET_KEEP_SESSIONS Function
- GET_MAX_SCHEDULER_JOBS Function
- GET_NO_PROXY_DOMAINS Function
- GET_OFFSET Function
- GET_PASS_ECID Function
- GET_PROXY Function
- GET_REMOTE_SERVER_AI_ATTRS Function
- GET_REMOTE_SERVER_AI_HEADERS Function
- GET_REMOTE_SERVER_AI_MODEL Function
- GET_REMOTE_SERVER_BASE_URL Function
- GET_REMOTE_SERVER_DEFAULT_DB Function
- GET_REMOTE_SERVER_HTTPS_HOST Function
- GET_REMOTE_SERVER_SQL_MODE Function
- GET_REST_SOURCE_CATALOG_GROUP Function
- GET_SCHEMA Function

- GET_THEME_ID Function
- GET_WORKSPACE_ID Function
- INSTALL Procedure
- REMOVE_APPLICATION Procedure
- SET_APPLICATION_ALIAS Procedure
- SET_APPLICATION_ID Procedure
- SET_APPLICATION_NAME Procedure
- SET_AUTHENTICATION_SCHEME Procedure
- SET_AUTO_INSTALL_SUP_OBJ Procedure
- SET_BUILD_STATUS Function
- SET_IMAGE_PREFIX Procedure
- SET_KEEP_BACKGROUND_EXECS Procedure
- SET_KEEP_SESSIONS Procedure
- SET_MAX_SCHEDULER_JOBS Procedure
- SET_OFFSET Procedure
- SET_PASS_ECID Procedure
- SET_PROXY Procedure
- SET_REMOTE_SERVER Procedure
- SET_REST_SOURCE_CATALOG_GROUP Procedure
- SET_SCHEMA Procedure
- SET_THEME_ID Procedure
- SET_WORKSPACE_ID Procedure
- SET_WORKSPACE Procedure
- SUSPEND_BACKGROUND_EXECS Procedure

8.1 About the APEX_APPLICATION_INSTALL API

Oracle APEX provides two ways to import an application into an APEX instance:

1. Uploading an application export file by using the web interface of APEX.
2. Execution of the application export file as a SQL script, typically in the command-line utility SQLcl.

Using the file upload capability of the web interface of APEX, 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 SQLcl, 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 APEX 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 APEX 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 APEX 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 are only used to override the default application installation behavior.

8.2 Attributes Manipulated by APEX_APPLICATION_INSTALL

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

Attribute	Description
Workspace ID	Workspace ID of the application to be imported. See GET_WORKSPACE_ID Function , SET_WORKSPACE_ID Procedure .
Application ID	Application ID of the application to be imported. See GENERATE_APPLICATION_ID Procedure , GET_APPLICATION_ID Function , SET_APPLICATION_ID Procedure .
Offset	Offset value used during application import. See GENERATE_OFFSET Procedure , GET_OFFSET Function , SET_OFFSET Procedure .
Schema	The parsing schema ("owner") of the application to be imported. See GET_SCHEMA Function , SET_SCHEMA Procedure .
Name	Application name of the application to be imported. See GET_APPLICATION_NAME Function , SET_APPLICATION_NAME Procedure .
Alias	Application alias of the application to be imported. See GET_APPLICATION_ALIAS Function , SET_APPLICATION_ALIAS Procedure .
Image Prefix	The image prefix of the application to be imported. See GET_IMAGE_PREFIX Function , SET_IMAGE_PREFIX Procedure .
Proxy	The proxy server attributes of the application to be imported. See GET_PROXY Function , SET_PROXY Procedure .

8.3 Import Data Types

The section describes import data types used by the APEX_APPLICATION_INSTALL package.

t_file_type

t_file_type data types define the kinds of install files.

```
subtype t_file_type is pls_integer range 1 .. 5;
c_file_type_workspace          constant t_file_type := 1;
c_file_type_app                constant t_file_type := 2;
c_file_type_websheet           constant t_file_type := 3;
```

```
c_file_type_plugin           constant t_file_type := 4;
c_file_type_css              constant t_file_type := 5;
```

 **Note:**

The constant `c_file_type_websheet` is no longer used in APEX and is obsolete.

t_app_usage

`t_app_usage` data types define the kinds of application usage.

```
subtype t_app_usage is pls_integer range 1..3;
c_app_usage_not_used      constant t_app_usage := 1;
c_app_usage_current_workspace constant t_app_usage := 2;
c_app_usage_other_workspace constant t_app_usage := 3;
```

t_file_info

`t_file_info` data types specify information in a source file that can be used to configure the installation.

```
type t_file_info is record (
    file_type                  t_file_type,
    workspace_id                number,
    version                     varchar2(10),
    app_id                      number,
    app_name                    varchar2(4000),
    app_alias                   varchar2(4000),
    app_owner                   varchar2(4000),
    build_status                varchar2(4000),
    has_install_script          boolean,
    app_id_usage                t_app_usage,
    app_alias_usage              t_app_usage );
```

8.4 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 SQLcl.

Import Application without Modification

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

```
@f645.sql
```

Import Application with Specified Application ID

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

```
BEGIN
    apex_application_install.set_application_id( 702 );
    apex_application_install.generate_offset;
    apex_application_install.set_application_alias( 'F' ||
apex_application_install.get_application_id );
END;
/
@645.sql
```

Import Application with Generated Application ID

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

```
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;
/
@f645.sql
```

Import Application into Different Workspace using Different Schema

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

```
BEGIN
    apex_application_install.set_workspace('FRED_PROD');
    apex_application_install.generate_offset;
    apex_application_install.set_schema( 'FREDDY' );
    apex_application_install.set_application_alias( 'FREDPROD_APP' );
END;
/
@f645.sql
```

Import into Training Instance for Three Different Workspaces

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

```
BEGIN
    apex_application_install.set_workspace('TRAINING1');
    apex_application_install.generate_application_id;
    apex_application_install.generate_offset;
    apex_application_install.set_schema( 'STUDENT1' );
```

```
    apex_application_install.set_application_alias( 'F' ||  
apex_application_install.get_application_id );  
END;  
/  
  
@f645.sql  
  
BEGIN  
    apex_application_install.set_workspace('TRAINING2');  
    apex_application_install.generate_application_id;  
    apex_application_install.generate_offset;  
    apex_application_install.set_schema( 'STUDENT2' );  
    apex_application_install.set_application_alias( 'F' ||  
apex_application_install.get_application_id );  
END;  
/  
  
@f645.sql  
  
BEGIN  
    apex_application_install.set_workspace('TRAINING3');  
    apex_application_install.generate_application_id;  
    apex_application_install.generate_offset;  
    apex_application_install.set_schema( 'STUDENT3' );  
    apex_application_install.set_application_alias( 'F' ||  
apex_application_install.get_application_id );  
    apex_application_install.end;  
    /  
  
@f645.sql
```

8.5 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;
```

8.6 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.

See Also:

- [GET_APPLICATION_ID Function](#)
- [Import Script Examples](#)
- [SET_APPLICATION_ID Procedure](#)

8.7 GENERATE_OFFSET Procedure

This procedure generates the offset value used during application import. Use the offset value to ensure that the metadata for the Oracle APEX 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 APEX generate this offset value for you.

Syntax

```
APEX_APPLICATION_INSTALL.GENERATE_OFFSET;
```

Parameters

None.

See Also:

- [GET_OFFSET Function](#)
- [Import Script Examples](#)
- [SET_OFFSET Procedure](#)

8.8 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 must be 255 characters or less.

```
DECLARE
    l_alias varchar2(255);
BEGIN
    l_alias := apex_application_install.get_application_alias;
END;
```

See Also:

[SET_APPLICATION_ALIAS Procedure](#)

8.9 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](#)

8.10 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](#)

8.11 GET_AUTHENTICATION_SCHEME Function

Use this function to retrieve the authentication scheme name that should override the default.

Syntax

```
APEX_APPLICATION_INSTALL.GET_AUTHENTICATION_SCHEME
    RETURN VARCHAR2
```

Example

Print the authentication scheme override.

```
select apex_application_install.get_authentication_scheme
    from sys.dual;
```

See Also:

[SET_AUTHENTICATION_SCHEME Procedure](#)

8.12 GET_AUTO_INSTALL_SUP_OBJ Function

This function retrieves the automatic install of supporting objects settings 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 is imported from the command line.

Syntax

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

Parameters

None.

Example

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

```
DECLARE
    l_auto_install_sup_obj boolean;
BEGIN
    l_auto_install_sup_obj :=
        apex_application_install.get_auto_install_sup_obj;
END;
```

 **See Also:**

[SET_AUTO_INSTALL_SUP_OBJ Procedure](#)

8.13 GET_BUILD_STATUS Function

This function retrieves the build status that overrides the default.

Syntax

```
APEX_APPLICATION_INSTALL.GET_BUILD_STATUS  
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example prints the build status override.

```
select apex_application_install.get_build_status  
from sys.dual;
```

 **See Also:**

[SET_BUILD_STATUS Function](#)

8.14 GET_IMAGE_PREFIX Function

This function gets the image prefix of the import application. Most Oracle APEX instances use the default image prefix of */i/*.

Syntax

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

Parameters

None.

Example

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

```
DECLARE
    l_image_prefix varchar2(255);
BEGIN
    l_image_prefix := apex_application_install.get_image_prefix;
END;
```

 **See Also:**

[SET_IMAGE_PREFIX Procedure](#)

8.15 GET_INFO Function

Use this function to retrieve install information from a source file.

Syntax

```
FUNCTION GET_INFO (
    p_source    IN apex_t_export_files )
RETURN t_file_info;
```

Parameters

Parameter	Description
p_source	The source code, a table of (name, contents) with a single record for normal APEX applications or multiple records for applications that were split when exporting. Note that passing multiple applications is not supported.

Returns

This function returns information about the application that can be used to configure the installation.

Raises

This function may raise the following: WWV_FLOW_IMP_PARSER.RUN_STMT_ERROR: The source contains invalid statements.

Example

The following example fetches an application from a remote URL and prints its install information.

```
DECLARE
    l_source apex_t_export_files;
    l_info    apex_application_install.t_file_info;
BEGIN
    l_source := apex_t_export_files (
        apex_t_export_file (
            name      => 'f100.sql',
            contents  => apex_web_service.make_rest_request (
                p_url          => 'https://
www.example.com/apps/f100.sql',
                p_http_method  => 'GET' ))));
    l_info   := apex_application_install.get_info (
        p_source      => l_source );
    sys.dbms_output.put_line (apex_string.format (
        p_message => q'!Type ..... %0
                           !Workspace ..... %1
                           !Version ..... %2
                           !App ID ..... %3
                           !App Name ..... %4
                           !Alias ..... %5
                           !Owner ..... %6
                           !Build Status ..... %7
                           !Has Install Script ... %8
                           !App ID Usage ..... %9
                           !App Alias Usage ..... %10!', 
        p0          => l_info.file_type,
        p1          => l_info.workspace_id,
        p2          => l_info.version,
        p3          => l_info.app_id,
        p4          => l_info.app_name,
        p5          => l_info.app_alias,
        p6          => l_info.app_owner,
        p7          => l_info.build_status,
        p8          => apex_debug.tochar(l_info.has_install_script),
        p9          => l_info.app_id_usage,
        p10         => l_info.app_alias_usage,
        p_prefix    => '!' ));
END;
```

See Also:

- [INSTALL Procedure](#)
- [GET_APPLICATION Function](#)

8.16 GET_KEEP_BACKGROUND_EXECS Function

This function checks if background executions are preserved or deleted during upgrades. Defaults to FALSE, so all background executions are aborted and deleted on application upgrade.

Syntax

```
APEX_APPLICATION_INSTALL.GET_KEEP_BACKGROUND_EXECS
    RETURN BOOLEAN;
```

Parameters

None.

Example

The following example shows whether background executions are preserved or deleted.

```
BEGIN
    dbms_output.put_line (
        CASE WHEN apex_application_install.get_keep_background_execs
            THEN 'background executions will be kept'
            ELSE 'background executions will be deleted'
        END );
END;
```

See Also:

- [SET_KEEP_BACKGROUND_EXECS Procedure](#)

8.17 GET_KEEP_SESSIONS Function

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

Syntax

```
function GET_KEEP_SESSIONS
    RETURN BOOLEAN
```

Example

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

```
dbms_output.put_line (
    case when apex_application_install.get_keep_sessions then 'sessions will
be kept'
```

```
else 'sessions will be deleted'  
end );
```

 **See Also:**

[SET_KEEP_SESSIONS Procedure](#)

8.18 GET_MAX_SCHEDULER_JOBS Function

This function fetches the maximum background processing jobs attribute during application import.

Syntax

```
APEX_APPLICATION_INSTALL.GET_MAX_SCHEDULER_JOBS  
RETURN NUMBER;
```

Parameters

None.

Example

```
DECLARE  
    l_max_scheduler_jobs number;  
BEGIN  
    l_max_scheduler_jobs := apex_application_install.get_max_scheduler_jobs;  
END;
```

 **See Also:**

- [SET_MAX_SCHEDULER_JOBS Procedure](#)

8.19 GET_NO_PROXY_DOMAINS Function

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

Syntax

```
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](#)

8.20 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](#)

8.21 GET_PASS_ECID Function

This function retrieves the pass ECID attribute value that overrides the default.

Syntax

```
APEX_APPLICATION_INSTALL.GET_PASS_ECID  
RETURN BOOLEAN;
```

Parameters

None.



[SET_PASS_ECID Procedure](#)

8.22 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 must be 255 characters or less.

```
DECLARE  
    l_proxy varchar2(255);  
BEGIN  
    l_proxy := apex_application_install.get_proxy;  
END;
```



[SET_PROXY Procedure](#)

8.23 GET_REMOTE_SERVER_AI_ATTRS Function

This function gets the AI attributes property to be used for a given remote server during application import.

Syntax

```
APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_AI_ATTRS (
    p_static_id IN VARCHAR2 )
    RETURN CLOB;
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object.

Example

```
DECLARE
    l_ai_attributes clob;
BEGIN
    l_ai_attributes :=
apex_application_install.get_remote_server_ai_attrs( 'MY_REMOTE_SERVER' );
END
```

 **See Also:**

[SET_REMOTE_SERVER Procedure](#)

8.24 GET_REMOTE_SERVER_AI_HEADERS Function

This function gets the AI HTTP Headers property to be used for a given remote server during application import.

Syntax

```
APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_AI_HEADERS (
    p_static_id IN VARCHAR2 )
    RETURN CLOB;
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object.

Example

```
DECLARE
    l_ai_http_headers clob;
BEGIN
    l_ai_http_headers :=
```

```
apex_application_install.get_remote_server_ai_headers( 'MY_REMOTE_SERVER' ) ;
END;
```

 **See Also:**

[SET_REMOTE_SERVER Procedure](#)

8.25 GET_REMOTE_SERVER_AI_MODEL Function

This function gets the AI model name property to be used for a given remote server during application import.

Syntax

```
APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_AI_MODEL (
    p_static_id IN VARCHAR2 )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object.

Example

```
DECLARE
    l_ai_model varchar2(255);
BEGIN
    l_ai_model :=
apex_application_install.get_remote_server_ai_model( 'MY_REMOTE_SERVER' );
END;
```

 **See Also:**

[SET_REMOTE_SERVER Procedure](#)

8.26 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

Parameter	Description
p_static_id	Static ID to reference the remote server object.

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](#)

8.27 GET_REMOTE_SERVER_DEFAULT_DB Function

This function gets the default database to be used for a given remote server during application import.

Syntax

```
APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_DEFAULT_DB (
    p_static_id IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object.

Example

```
DECLARE
    l_default_database varchar2(255);
BEGIN
    l_default_database :=
```

```
apex_application_install.get_remote_server_default_db( 'MY_REMOTE_SERVER' );
END;
```

**Note:**

[SET_REMOTE_SERVER Procedure](#)

8.28 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

Parameter	Description
p_static_id	Static ID to reference the remote server object.

Example

```
DECLARE
    l_https_host varchar2(255);
BEGIN
    l_https_host :=
apex_application_install.get_remote_server_https_host( 'MY_REMOTE_SERVER' );
END;
```

**See Also:**

[SET_REMOTE_SERVER Procedure](#)

8.29 GET_REMOTE_SERVER_SQL_MODE Function

This function gets the SQL mode to be used for a given remote MySQL server during application import.

Syntax

```
APEX_APPLICATION_INSTALL.GET_REMOTE_SERVER_SQL_MODE (
    p_static_id IN VARCHAR2 )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object..

Example

```
DECLARE
    l_default_database varchar2(255);
BEGIN
    l_default_database :=
apex_application_install.get_remote_server_sql_mode( 'MY_REMOTE_SERVER' );
END;
```

 **Note:**

[SET_REMOTE_SERVER Procedure](#)

8.30 GET_REST_SOURCE_CATALOG_GROUP Function

This function retrieves the name of REST Source Catalog Group which new catalogs are imported into.

Syntax

```
APEX_APPLICATION_INSTALL.GET_REST_SOURCE_CATALOG_GROUP
    RETURN VARCHAR2;
```

Parameters

None.

Example

The following example prints the REST Source Catalog Group override.

```
select apex_application_install.get_rest_source_catalog_group
    from sys.dual;
```

 **Note:**

[SET_REST_SOURCE_CATALOG_GROUP Procedure](#)

8.31 GET_SCHEMA Function

Use this function to get the parsing schema (owner) of the APEX application.

Syntax

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

Parameters

None.

Example

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

```
DECLARE  
    l_schema varchar2(30);  
BEGIN  
    l_schema := apex_application_install.get_schema;  
END;
```

 **See Also:**

[SET_SCHEMA Procedure](#)

8.32 GET_THEME_ID Function

This function retrieves the Theme ID value that overrides the default.

Syntax

```
APEX_APPLICATION_INSTALL.GET_THEME_ID  
RETURN NUMBER
```

Parameters

None.

Returns

This function returns the Theme ID value.

Example

The following example prints the theme ID override.

```
select apex_application_install.get_theme_id from sys.dual
```



[SET_THEME_ID Procedure](#)

8.33 GET_WORKSPACE_ID Function

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

Syntax

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

Parameters

None.

Example

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

```
DECLARE
    l_workspace_id number;
BEGIN
    l_workspace_id := apex_application_install.get_workspace_id;
END;
```



[SET_WORKSPACE_ID Procedure](#)

8.34 INSTALL Procedure

This procedure installs an application. Use the APEX_APPLICATION_INSTALL.SET% procedures to configure installation parameters.

Syntax

```
PROCEDURE INSTALL (
    p_source          IN apex_t_export_files      DEFAULT NULL,
    p_overwrite_existing IN BOOLEAN                DEFAULT FALSE );
```

Parameters

Parameter	Description
p_source	The source code, a table of (name, contents) with a single record for normal Oracle APEX applications or multiple records for applications that were split when exporting. Passing multiple applications is not supported. If null (default), imports the source that was previously passed to GET_INFO.
p_overwrite_existing	If FALSE (default), raises an error instead of overwriting an existing application.

Raises

- WWV_FLOW_IMP_PARSER, RUN_STMT_ERROR: The source contains invalid statements.
- SECURITY_GROUP_ID_INVALID: The current workspace conflicts with the install workspace.
- WWV_FLOW_API.FLOW_ID_RESERVED_FOR_OTHER_WORKSPACE: The application ID is used in another workspace.
- WWV_FLOW_API.FLOW_ID_RANGE_RESERVED: The application ID is reserved for internal use.
- WWV_FLOW_API.FLOW_ID_OUT_OF_RANGE: The application ID used for installing is not in a valid range.
- APPLICATION_ID_RESERVED: The application ID is in use in the current workspace and p_overwrite_existing was set to false.

Example

Fetch an application from a remote URL, then install it with a new ID and new component ID offsets in workspace EXAMPLE.

```
DECLARE
    l_source apex_t_export_files;
    l_info    apex_application_install.t_file_info;
BEGIN
    l_source := apex_t_export_files (
        apex_t_export_file (
            name      => 'f100.sql',
            contents  => apex_web_service.make_rest_request (
                p_url      => 'https://
www.example.com/apps/f100.sql',
                p_http_method => 'GET' )));

    apex_util.set_workspace('EXAMPLE');
    apex_application_install.generate_application_id;
    apex_application_install.generate_offset;
    apex_application_install.install (
```

```
    p_source => l_source );
END;
```

8.35 REMOVE_APPLICATION Procedure

This procedure removes an application from a workspace. Use the APEX_APPLICATION_INSTALL.SET_% procedures to configure parameters.

Syntax

```
APEX_APPLICATION_INSTALL.REMOVE_APPLICATION (
    p_application_id IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	The ID of the application.

Raises

This procedure may raise the following:

- WWV_FLOW_API.DELETE_APP_IN_DIFFERENT_WORKSPACE: The application is not in this workspace.
- WWV_FLOW_API.FLOW_NOT_DELETED: The application was not deleted.
- WWV_FLOW_APP_NOT_FOUND_ERR: The application ID was not found.

Example

The following example demonstrates how to use the REMOVE_APPLICATION procedure to remove an application with an ID of 100 from a workspace.

```
BEGIN
    apex_application_install.set_workspace('EXAMPLE');
    apex_application_install.set_keep_sessions(false);
    apex_application_install.remove_application(100);
END;
```

8.36 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

Parameter	Description
p_application_alias	The application alias. The application alias is an alphanumeric identifier. Must be fewer than 255 characters and unique within a workspace. (Optional) Oracle recommends that the alias be unique within an entire instance.

 **See Also:**

- [GET_APPLICATION_ALIAS Function](#)
- [Import Script Examples](#)

8.37 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 Oracle APEX application IDs.

Syntax

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

Parameters

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

 **See Also:**

- [SET_APPLICATION_ID Procedure](#)
- [Import Script Examples](#)
- [GENERATE_APPLICATION_ID Procedure](#)

8.38 SET_APPLICATION_NAME Procedure

This procedure sets the application name of the application to be imported.

Syntax

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

Parameters

Parameter	Description
p_application_name	This is the application name. The application name cannot be null and must be 255 characters or less.

Example

The following example sets the application name for app 100 to "Executive Dashboard".

```
BEGIN
    apex_application_install.set_application_name( p_application_name =>
'Executive Dashboard' );
END;
/
@f100.sql
```

 **See Also:**

[GET_APPLICATION_NAME Function](#)

8.39 SET_AUTHENTICATION_SCHEME Procedure

Use this procedure to override the active authentication scheme for the applications that are about to be installed.

Syntax

```
APEX_APPLICATION_INSTALL.SET_AUTHENTICATION_SCHEME (
    p_name IN VARCHAR2 )
```

Parameters

Parameter	Description
p_name	The name of the authentication scheme to be activated. This new authentication scheme must exist in the application. If NULL, the active authentication scheme will remain unchanged.

Example

The following example activates authentication scheme "SSO-Production" and installs application f100.sql, then resets the override for f101.sql to keep its active scheme.

```
BEGIN
    apex_application_install.set_authentication_scheme (
        p_name => 'SSO-Production' );
END;
/
@f100.sql
BEGIN
    apex_application_install.set_authentication_scheme (
        p_name => null );
END;
/
@f101.sql
```

 **See Also:**

[GET_AUTHENTICATION_SCHEME Function](#)

8.40 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 is imported from the command line.

Syntax

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

Parameters

Parameter	Description
p_auto_install_sup_obj	Boolean value for the automatic install of supporting objects.

Example

The following example enables the automatic installation of supporting objects for app 100.

```
BEGIN
    apex_application_install.set_auto_install_sup_obj( p_auto_install_sup_obj
=> true );
END;
```

```
/  
@f100.sql
```

 **See Also:**

[GET_AUTO_INSTALL_SUP_OBJ Function](#)

8.41 SET_BUILD_STATUS Function

Use this function to override the build status for applications that are about to be installed.

Syntax

```
APEX_APPLICATION_INSTALL.SET_BUILD_STATUS (  
    p_build_status    IN    wvv_flow_application_admin_api.t_build_status )
```

Parameters

Parameter	Description
p_build_status	New build status to set the application to. Values include: <ul style="list-style-type: none">• apex_application_admin.c_build_status_run_and_build - Developers and users can both run and develop the application.• apex_application_admin.c_build_status_run_only - Only users can run the application. Developers cannot edit the application.

Example

The following example sets build status for app 100 to RUN_ONLY.

```
BEGIN  
    apex_application_install.set_build_status (   
        p_build_status => 'RUN_ONLY' );  
END;  
/  
@f100.sql
```

 **See Also:**

[GET_BUILD_STATUS Function](#)

8.42 SET_IMAGE_PREFIX Procedure

This procedure sets the image prefix of the import application. Most Oracle APEX instances use the default image prefix of /i/.

Syntax

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

Parameters

Parameter	Description
p_image_prefix	The image prefix. It can be a fully qualified domain, like a CDN or another web server, or just a path.

Example

The following example sets the value of the image prefix attribute for app 100 to /i/

```
begin
    apex_application_install.set_image_prefix( p_image_prefix => '/i/' );
end;
/
@f100.sql
```

 **See Also:**

[GET_IMAGE_PREFIX Function](#)

8.43 SET_KEEP_BACKGROUND_EXECS Procedure

This procedure preserves background executions associated with the application during upgrades.

Syntax

```
APEX_APPLICATION_INSTALL.SET_KEEP_BACKGROUND_EXECS (
    p_keep_background_execs IN BOOLEAN )
```

Parameters

Parameter	Description
p_keep_background_execs	TRUE to preserve background executions. FALSE to delete them.

Example

The following example installs application 100 in workspace FRED_PROD and preserves background executions.

```
BEGIN
    apex_application_install.set_workspace(p_workspace => 'FRED_PROD');
    apex_application_install.set_keep_background_execs(p_keep_background_execs
=> true);
END;
/
@f100.sql
```

 **See Also:**

- [GET_KEEP_BACKGROUND_EXECS Function](#)

8.44 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

Parameter	Description
p_keep_sessions	Default FALSE. If FALSE, sessions are deleted. If TRUE, sessions are preserved. KEEP_SESSIONS_ON_UPGRADE controls the default behavior. If N (default), sessions are deleted. KEEP_SESSIONS_ON_UPGRADE is an instance parameter.

Example

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

```
BEGIN
    apex_application_install.set_workspace(p_workspace => 'FRED_PROD');
    apex_application_install.set_keep_sessions(p_keep_sessions => true);
END;
/
@f100.sql
```

 See Also:

- [GET_KEEP_SESSIONS Function](#)

8.45 SET_MAX_SCHEDULER_JOBS Procedure

This procedure sets the maximum background processing jobs attribute of the application to be imported.

Syntax

```
APEX_APPLICATION_INSTALL.SET_MAX_SCHEDULER_JOBS (
    p_max_scheduler_jobs IN NUMBER )
```

Parameters

Parameter	Description
p_max_scheduler_jobs	Maximum number of background processing jobs for the application to be imported.

Example

The following example sets the maximum number of background processing jobs for app 100 to 5.

```
BEGIN
    apex_application_install.set_max_scheduler_jobs(
        p_max_scheduler_jobs => 5 );
END;
/
@f100.sql
```

 See Also:

- [GET_MAX_SCHEDULER_JOBS Function](#)

8.46 SET_OFFSET Procedure

This procedure sets the offset value used during application import. Use the offset value to ensure that the metadata for the Oracle APEX 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 use APEX to automatically generate this offset value for you.

Syntax

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

Parameters

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

Example

The following example generates a random number from the database and uses this as the offset value for app 100.

```
DECLARE
    l_offset number;
BEGIN
    l_offset := dbms_random.value(100000000000, 999999999999);
    apex_application_install.set_offset( p_offset => l_offset );
END;
/
@f100.sql
```

See Also:

- [GET_OFFSET Function](#)
- [GENERATE_OFFSET Procedure](#)

8.47 SET_PASS_ECID Procedure

This procedure overrides the pass Execution Context ID (ECID) attribute for applications that are being installed.

Syntax

```
APEX_APPLICATION_INSTALL.SET_PASS_ECID (
    p_pass_ecid IN BOOLEAN )
```

Parameters

Parameter	Description
p_pass_ecid	New pass ECID value to set application to. Values include: <ul style="list-style-type: none"> • TRUE: Pass the ECID to the external web services for end-to-end tracing. • FALSE: Deny the ECID.

Example

The following example sets Pass ECID to true.

```
BEGIN
    apex_application_install.set_pass_ecid (
        p_pass_ecid => true );
END;
/
@f100.sql
```

 **See Also:**

[GET_PASS_ECID Function](#)

8.48 SET_PROXY Procedure

This procedure sets the proxy server attributes of an imported application.

Syntax

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

Parameters

Parameter	Description
p_proxy	The proxy server. There is no default value. The proxy server must be fewer than 255 characters and must exclude any protocol prefix (such as <code>http://</code>). The following is a valid example: <code>www-proxy.example.com</code>
p_no_proxy_domains	Default null. The list of domains for which the proxy server can not be used.

Example

The following example sets the value of the proxy attribute for app 100 to www-proxy.example.com.

```
BEGIN
  apex_application_install.set_proxy( p_proxy => 'www-proxy.example.com' );
END;
/
@f100.sql
```

 **See Also:**

[GET_PROXY Function](#)

[GET_NO_PROXY_DOMAINS Function](#)

8.49 SET_REMOTE_SERVER Procedure

This procedure sets 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,
  --
  p_default_database IN VARCHAR2 DEFAULT NULL,
  p_mysql_sql_modes IN VARCHAR2 DEFAULT NULL,
  --
  p_ordes_timezone IN VARCHAR2 DEFAULT NULL,
  --
  p_ai_model_name  IN VARCHAR2 DEFAULT NULL,
  p_ai_http_headers IN CLOB      DEFAULT NULL,
  p_ai_attributes   IN CLOB      DEFAULT NULL )
```

Parameters

Parameter	Description
p_static_id	Static ID to reference the remote server object.
p_base_url	New Base URL to use for this remote server object.
p_https_host	New HTTPS Host Property to use for this remote server object. Only relevant when the base URL is https:// and the Oracle Database version is 12.2 or greater.
p_default_database	Default database to use when connecting. Currently only supported for MySQL databases.

Parameter	Description
p_mysql_sql_modes	SQL modes to use when connecting to a MySQL database.
p_ords_timezone	Time zone in which the ORDS server of a REST-enabled SQL reference used by the application runs.
p_ai_model_name	The AI model to use when requesting a response from a Generative AI Service.
p_ai_http_headers	HTTP headers to use when making a request to a Generative AI Service.
p_ai_attributes	Attributes in JSON format to use when making a request to a Generative AI Service.

Example

The following example sets the Base URL attribute of the remote server MY_REMOTE_SERVER for app 100.

```
BEGIN
    apex_application_install.set_remote_server(
        p_static_id => 'MY_REMOTE_SERVER',
        p_base_url => 'http://production.example.com' );
END;
```

See Also:

- [GET_REMOTE_SERVER_BASE_URL Function](#)
- [GET_REMOTE_SERVER_HTTPS_HOST Function](#)
- [GET_REMOTE_SERVER_DEFAULT_DB Function](#)
- [GET_REMOTE_SERVER_SQL_MODE Function](#)
- [GET_REMOTE_SERVER_AI_MODEL Function](#)
- [GET_REMOTE_SERVER_AI_HEADERS Function](#)
- [GET_REMOTE_SERVER_AI_ATTRS Function](#)

8.50 SET_REST_SOURCE_CATALOG_GROUP Procedure

This procedure sets the REST Source Catalog group to import a new REST Source Catalog.

Syntax

```
APEX_APPLICATION_INSTALL.SET_REST_SOURCE_CATALOG_GROUP (
    p_name   IN VARCHAR2 )
```

Parameters

Parameter	Description
p_name	The name of the REST Source Catalog Group. That Group must exist in the workspace.

Example

The following example sets the catalog group to Financial Services Catalogs. REST Source Catalogs are imported into this group.

```
BEGIN
    apex_application_install.set_rest_source_catalog_group (
        p_name => 'Financial Services Catalogs' );
END;
/
@rest-service-catalog-financial.sql
```

 **Note:**

[GET_REST_SOURCE_CATALOG_GROUP Function](#)

8.51 SET_SCHEMA Procedure

Use this function to set the parsing schema (owner) of the Oracle APEX 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

Parameter	Description
p_schema	The schema name.

 **See Also:**

- [GET_SCHEMA Function](#)
- [Import Script Examples](#)

8.52 SET_THEME_ID Procedure

This procedure overrides the Theme ID attribute for Template Components that are about to be installed.

Syntax

```
APEX_APPLICATION_INSTALL.SET_THEME_ID (
    p_theme_id    IN NUMBER )
```

Parameters

Parameter	Description
p_theme_id	New Theme ID value to install the Template Component.

Example

The following example sets "Theme ID" to 42.

```
BEGIN
    apex_application_install.set_theme_id (
        p_theme_id => 42 );
END;
/
@plugin.sql
```

See Also:

[GET_THEME_ID Function](#)

8.53 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

Parameter	Description
p_workspace_id	The workspace ID.

 See Also:

- [SET_WORKSPACE_ID Procedure](#)
- [Import Script Examples](#)

8.54 SET_WORKSPACE Procedure

This procedure sets the workspace ID for an application to be imported.

Syntax

```
APEX_APPLICATION_INSTALL.SET_WORKSPACE (
    p_workspace IN VARCHAR2 );
```

Parameters

Parameters	Description
p_workspace	The workspace name.

Example

The following example sets the workspace ID for app 100 to workspace "FRED_PROD".

```
BEGIN
    apex_application_install.set_workspace (
        p_workspace => 'FRED_PROD' );
END;
/
@f100.sql
```

 See Also:

- [GET_WORKSPACE_ID Function](#)
- [SET_WORKSPACE_ID Procedure](#)

8.55 SUSPEND_BACKGROUND_EXECS Procedure

This procedure suspends background page processing for an application. This procedure is intended for use before upgrades.

This procedure enables orderly application upgrades by waiting for all SCHEDULED or EXECUTING background executions to complete then locking out subsequent processes until after the upgrade. During the time when background executions are suspended for an application, new executions can be enqueued, but are not executed, until the lock releases.

The lock releases when the transaction ends with a COMMIT or ROLLBACK operation.

Syntax

```
APEX_APPLICATION_INSTALL.SUSPEND_BACKGROUND_EXECS (  
    p_application_id      IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	The application ID.

Example

```
BEGIN  
    apex_application_install.suspend_background_execs(  
        p_application_id => 100 );  
END;
```

APEX_APPROVAL (Deprecated)

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

The APEX_APPROVAL package provides APIs for the management of approvals and Human Tasks. This package includes functionality to create new Human Tasks for a user to approve as well as operations dealing with the lifecycle management and state handling of Human Tasks. This package is part of the Oracle APEX Workflow functionality.

- [Constants and Data Types](#)
- [ADD_TASK_COMMENT Procedure](#)
- [ADD_TASK_POTENTIAL_OWNER Procedure](#)
- [ADD_TO_HISTORY Procedure](#)
- [APPROVE_TASK Procedure](#)
- [CANCEL_TASK Procedure](#)
- [CLAIM_TASK Procedure](#)
- [COMPLETE_TASK Procedure](#)
- [CREATE_TASK Function](#)
- [DELEGATE_TASK Procedure](#)
- [GET_LOV_PRIORITY Function](#)
- [GET_LOV_STATE Function](#)
- [GET_NEXT_PURGE_TIMESTAMP Function](#)
- [GET_TASK_DELEGATES Function](#)
- [GET_TASK_HISTORY Function](#)
- [GET_TASK_PARAMETER_OLD_VALUE Function](#)
- [GET_TASK_PARAMETER_VALUE Function](#)
- [GET_TASK_PRIORITIES Function](#)
- [GET_TASKS Function](#)
- [HANDLE_TASK_DEADLINES Procedure](#)
- [HAS_TASK_PARAM_CHANGED Function](#)
- [IS_ALLOWED Function](#)
- [IS_BUSINESS_ADMIN Function](#)
- [IS_OF_PARTICIPANT_TYPE Function](#)

- REJECT_TASK Procedure
- RELEASE_TASK Procedure
- REMOVE_POTENTIAL_OWNER Procedure
- RENEW_TASK Function
- REQUEST_MORE_INFORMATION Procedure
- SET_INITIATOR_CAN_COMPLETE Procedure
- SET_TASK_DUE Procedure
- SET_TASK_PARAMETER_VALUES Procedure
- SET_TASK_PRIORITY Procedure
- SUBMIT_INFORMATION Procedure

9.1 Constants and Data Types

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

The APEX_APPROVAL package uses the following constants and data types.

Task Types

```
c_task_type_approval      constant t_task_type := 'APPROVAL';
c_task_type_action        constant t_task_type := 'ACTION';
```

Task List Context Types

```
c_context_my_tasks        constant t_task_list_context := 'MY_TASKS';
c_context_admin_tasks      constant t_task_list_context := 'ADMIN_TASKS';
c_context_initiated_by_me  constant t_task_list_context :=
'INITIATED_BY_ME';
c_context_single_task      constant t_task_list_context := 'SINGLE_TASK';
```

Task Definition Participant Types

```
c_task_potential_owner    constant t_task_participant_type :=
'POTENTIAL_OWNER';
c_task_business_admin      constant t_task_participant_type :=
'BUSINESS_ADMIN';
```

Task Definition Participant Identity Types

```
c_task_identity_type_user  constant t_task_identity_type := 'USER';
```

Task (Instance) Priority Constants

```
c_task_priority_lowest      constant integer := 5;
c_task_priority_low        constant integer := 4;
c_task_priority_medium    constant integer := 3;
c_task_priority_high       constant integer := 2;
c_task_priority_urgent     constant integer := 1;
```

Task (Instance) States

```
c_task_state_unassigned    constant t_task_state := 'UNASSIGNED';
c_task_state_assigned      constant t_task_state := 'ASSIGNED';
c_task_state_completed     constant t_task_state := 'COMPLETED';
c_task_state_cancelled    constant t_task_state := 'CANCELLED';
c_task_state_failed        constant t_task_state := 'FAILED';
c_task_state_errorred      constant t_task_state := 'ERRORED';
c_task_state_expired       constant t_task_state := 'EXPIRED';
c_task_state_info_requested constant t_task_state := 'INFO_REQUESTED';
```

Task (Instance) Outcomes

```
c_task_outcome_approved   constant t_task_outcome := 'APPROVED';
c_task_outcome_rejected    constant t_task_outcome := 'REJECTED';
```

Task (Instance) Operations

```
c_task_op_approve         constant t_task_operation := 'APPROVE_TASK';
c_task_op_reject           constant t_task_operation := 'REJECT_TASK';
c_task_op_complete         constant t_task_operation := 'COMPLETE_TASK';
c_task_op_claim            constant t_task_operation := 'CLAIM_TASK';
c_task_op_delegate         constant t_task_operation := 'DELEGATE_TASK';
c_task_op_renew             constant t_task_operation := 'RENEW_TASK';
c_task_op_release           constant t_task_operation := 'RELEASE_TASK';
c_task_op_cancel            constant t_task_operation := 'CANCEL_TASK';
c_task_op_set_priority     constant t_task_operation := 'SET_TASK_PRIORITY';
c_task_op_add_comment      constant t_task_operation := 'ADD_TASK_COMMENT';
c_task_op_add_owner         constant t_task_operation :=
'ADD_TASK_POTENTIAL_OWNER';
c_task_op_request_info     constant t_task_operation := 'REQUEST_INFO';
c_task_op_submit_info       constant t_task_operation := 'SUBMIT_INFO';
c_task_op_set_due_date     constant t_task_operation := 'SET_DUE_DATE';
c_task_op_remove_owner      constant t_task_operation :=
'REMOVE_POTENTIAL_OWNER';
c_task_op_set_params        constant t_task_operation := 'SET_TASK_PARAMS';
```

Task (Instance) date formats

```
c_canonical_date_format   constant varchar2(16)      := 'YYYYMMDDHH24MISS';
```

Task Parameters Default

```
c_empty_task_parameters t_task_parameters;
```

Global Data Types

```
subtype t_task_participant_type is varchar2(15);
subtype t_task_identity_type    is varchar2(32);
subtype t_task_type             is varchar2(32);
subtype t_task_outcome          is varchar2(32);
subtype t_task_state            is varchar2(15);
subtype t_task_operation         is varchar2(30);
subtype t_task_list_context      is varchar2(15);
```

Data Types**Task Parameter (Value)**

Attribute	Description
static_id	The static ID of the parameter. This ID must match the static ID of the corresponding parameter in the task definition.
string_value	The value of the parameter as a string.

```
type t_task_parameter is record (
    static_id          varchar2(32767),
    string_value        varchar2(32767)
);
```

Collection of Task Parameter Values

```
type t_task_parameters is table of t_task_parameter index by pls_integer;
```

Collection of Task Participant Types

```
type t_task_participant_types is table of t_task_participant_type
index by pls_integer;
```

9.2 ADD_TASK_COMMENT Procedure

Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure adds a comment to a task. Any potential owner or business administrator of a Task can add comments to a Task. Comments are useful as additional information regarding a Task. For example, a manager may add her notes to a Task she is working on before delegating the Task.

Syntax

```
APEX_APPROVAL.ADD_TASK_COMMENT (
    p_task_id           IN NUMBER,
    p_text              IN VARCHAR2 );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_text	The comment text.

Example

```
BEGIN
    add_task_comment(
        p_task_id => 1234,
        p_text      => 'Please review and approve');
END;
```

9.3 ADD_TASK_POTENTIAL_OWNER Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure adds a new potential owner to a task. Only a Business Administrator for the task can invoke this procedure. The procedure throws an error if the task is in Completed or Errorred state.

Syntax

```
APEX_APPROVAL.ADD_TASK_POTENTIAL_OWNER (
    p_task_id           IN NUMBER,
    p_potential_owner   IN VARCHAR2,
    p_identity_type     IN t_task_identity_type default
    c_task_identity_type_user );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_potential_owner	The potential owner.
p_identity_type	The identity type of the potential owner. Default is USER.

 **Note:**

As of this release, the only supported identity type is USER. Additional options will be added in a future release.

Example

The following example adds user STIGER as potential owner for Task ID 1234.

```
BEGIN
    apex_approval.add_task_potential_owner(
        p_task_id      => 1234,
        p_potential_owner => 'STIGER'
    );
END;
```

9.4 ADD_TO_HISTORY Procedure

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure adds a log entry into the task history and is to be used within task action code.

Syntax

```
APEX_APPROVAL.ADD_TO_HISTORY (
    p_message IN VARCHAR2 )
```

Parameters

Parameter	Description
p_message	Message to add into to the task history.

Example

The following example demonstrates how to write log information. The task action uses `select * from emp` as the action source query.

```
BEGIN
    apex_approval.add_to_history(
        p_message => 'Approved leave for employee with empno: ' || :EMPNO );
    my_logic_package.update_emp_leave_balance(
        p_empno      => :EMPNO,
        p_no_of_days  => :NO_OF_DAYS);
END;
```

9.5 APPROVE_TASK Procedure

Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure approves a Task. Only the potential owner or actual owner of the task can invoke this procedure. This procedure moves the state of the Task to `Completed` and sets the outcome of the Task to `Approved`.

This is a convenience procedure and equivalent to calling `complete_task` with outcome `apex_approval.c_task_outcome_approved`.

See Also:

[COMPLETE_TASK Procedure](#)

Syntax

```
APEX_APPROVAL.APPROVE_TASK (
    p_task_id          IN NUMBER,
    p_autoclaim        IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
<code>p_task_id</code>	The Task ID.
<code>p_autoclaim</code>	If Task is in state <code>UNASSIGNED</code> then claim the task implicitly.

State Handling

Pre-State: `ASSIGNED | UNASSIGNED` (`p_autoclaim=true`)

Post-State: COMPLETED

Example

```
BEGIN
    apex_approval.approve_task(
        p_task_id => 1234);
END;
```

9.6 CANCEL_TASK Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure cancels the task by setting the task to state CANCELED. Only the initiator or the Business Administrator of the task can invoke this procedure. Only tasks which are not in COMPLETED or ERRORED state can be CANCELED.

Canceling a task is useful when an approval is no longer required. For example, consider a travel approval for a business trip, and the person requesting the approval suddenly cannot make the trip, and the Task may be canceled.

Syntax

```
APEX_APPROVAL.CANCEL_TASK (
    p_task_id           IN NUMBER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.

State Handling

Pre-State: Any

Post-State: CANCELLED

Example

```
BEGIN
    apex_approval.cancel_task(
        p_task_id => 1234
    );
END;
```

9.7 CLAIM_TASK Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure claims responsibility for a task. A task can be claimed by potential owners of the Task. A Task must be in Unassigned state to claim it. Once the task is claimed by a user, the Task transitions to Assigned state and the actual owner of the task is set to the user who claimed the task.

Syntax

```
APEX_APPROVAL.CLAIM_TASK (
    p_task_id           IN NUMBER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.

State Handling

Pre-State: UNASSIGNED. Post-State: ASSIGNED.

Example

```
BEGIN
    apex_approval.claim_task(
        p_task_id => 1234);
END;
```

9.8 COMPLETE_TASK Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure completes a task. For Approval Tasks, an outcome must be supplied (Approved or Rejected). Action Tasks do not have an outcome. Only the actual owner or a potential owner of the task can invoke this procedure.

Tasks in Assigned state might be completed with an outcome. This operation transitions the Task from Assigned state to Completed state and sets the outcome of the task. Once a Task is in Completed state, it is subject for purging and archival.

Syntax

```
APEX_APPROVAL.COMPLETE_TASK (
    p_task_id          IN NUMBER,
    p_outcome          IN t_task_outcome DEFAULT NULL,
    p_autoclaim        IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_outcome	The outcome of the Task for Approval Tasks.
p_autoclaim	If Task is in state UNASSIGNED then claim the task implicitly.

State Handling

Pre-State: ASSIGNED | UNASSIGNED (p_autoclaim=true)

Post-State: COMPLETED

Example

```
BEGIN
    apex_approval.complete_task(
        p_task_id => 1234,
        p_outcome => apex_approval.c_task_outcome_approved
    );
END;
```

9.9 CREATE_TASK Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function creates a new task. A new Task (Instance) is created. Depending on the task definition participant setting, the Task is set to state Unassigned or Assigned.

If the task definition has a single potential owner, the Task is set to Assigned.

If the task has multiple potential owners, the Task is set to Unassigned and can be claimed by any of the potential owners. This procedure throws an exception if no potential owners are found in the corresponding task definition.

Syntax

```
APEX_APPROVAL.CREATE_TASK (
    p_application_id      IN NUMBER          DEFAULT
apex_application.g_flow_id,
    p_task_def_static_id  IN VARCHAR2,
    p_subject              IN VARCHAR2        DEFAULT NULL,
    p_parameters            IN t_task_parameters DEFAULT
c_empty_task_parameters,
    p_priority              IN INTEGER         DEFAULT NULL,
    p_initiator             IN VARCHAR2        DEFAULT NULL,
    p_initiator_can_complete IN BOOLEAN        DEFAULT NULL,
    p_detail_pk             IN VARCHAR2        DEFAULT NULL,
    p_due_date               IN TIMESTAMP WITH TIME ZONE DEFAULT NULL )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_application_id	The application ID that creates the Task.
p_task_def_static_id	The Task Definition static ID.
p_subject	The subject (expression of the Task).
p_parameters	The task parameters.
p_priority	(Optional) A task priority, default is NULL. If no priority is provided, uses the priority set in the corresponding task definition.
p_initiator	(Optional) If TRUE, the initiator of the task instance can approve or reject this task. If no value is provided, the setting in the corresponding task definition is used.
p_initiator_can_complete	(Optional) Enables the initiator of a task to complete the task (default NULL). If this parameter is not specified, the value of the corresponding task definition is used.
p_detail_pk	(Optional) A primary key value for the task details.
p_due_date	(Optional) Page Item representing the Due Date of the Task. When specified, this value overrides the Due Date provided in the Task Definition this Task is based on.

Returns

Returns the ID of the newly created task.

Example

The following example creates a requisition item in the system of record in the database and then creates a new Human Task to get the requisition item approved by a user.

```
DECLARE
    l_req_id      number;
    l_req_item    varchar2(100) := 'Some requisition item requiring approval';
    l_req_amount   number := 2499.42;
    l_task_id     number;
BEGIN
    insert into requisitions(created_by, creator_emailid, item, item_amount,
```

```

item_category)
    values (:emp_uid, :emp_email, l_req_item, l_req_amount, 'Equipment')
    returning id into l_req_id;
commit;

l_task_id := apex_approval.create_task(
    p_application_id => 110,
    p_task_def_static_id => 'REQAPPROVALS',
    p_subject => 'Requisition ' || l_req_id || ':' || l_req_item || '
for ' || l_req_amount,
    p_initiator => :emp_uid,
    p_initiator_can_complete => true,
    p_parameters => apex_approval.t_task_parameters(
        1 => apex_approval.t_task_parameter(static_id => 'REQ_DATE',
string_value => sysdate),
        3 => apex_approval.t_task_parameter(static_id => 'REQ_AMOUNT',
string_value => l_req_amount),
        4 => apex_approval.t_task_parameter(static_id => 'REQ_ITEM',
string_value => l_req_item),
        5 => apex_approval.t_task_parameter(static_id => 'REQ_ID',
string_value => l_req_id)
    ),
    p_detail_pk => l_req_id
);
END;

```

9.10 DELEGATE_TASK Procedure

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure assigns the task to one potential owner and sets the task state to Assigned. Either the current owner of the task (the user to whom the task is currently assigned) or the Business Administrator of the task can perform this operation.

Syntax

```
APEX_APPROVAL.DELEGATE_TASK (
    p_task_id          IN NUMBER,
    p_to_user          IN VARCHAR2 );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_to_user	A (user) participant.

State Handling

Pre-State: UNASSIGNED, ASSIGNED

Post-State: ASSIGNED

Example

```
BEGIN
    apex_approval.delegate_task(
        p_task_id          => 1234,
        p_to_user          => 'STIGER'
    );
END;
```

9.11 GET_LOV_PRIORITY Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function retrieves the list of value data for the task priority.

Syntax

```
APEX_APPROVAL.GET_LOV_PRIORITY
RETURN apex_t_temp_lov_data pipelined;
```

Returns

A table of `apex_t_temp_lov_data`.

Example

```
select disp,val from table ( apex_approval.get_lov_priority )
```

9.12 GET_LOV_STATE Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function gets the list of value data for the task attribute state.

Syntax

```
APEX_APPROVAL.GET_LOV_STATE  
RETURN apex_t_temp_lov_data pipelined;
```

Returns

A table of `apex_t_temp_lov_data`.

Example

```
select disp,val from table ( apex_approval.get_lov_state )
```

9.13 GET_NEXT_PURGE_TIMESTAMP Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function retrieves the timestamp of the next purge.

Syntax

```
APEX_APPROVAL.GET_NEXT_PURGE_TIMESTAMP  
RETURN timestamp with time zone;
```

Parameters

None.

Returns

Returns the timestamp of the next purge.

Example

```
DECLARE  
    l_next_purge_job_ts timestamp with time zone;  
BEGIN  
    l_next_purge_job_ts := apex_approval.get_next_purge_timestamp();  
END;
```

9.14 GET_TASK_DELEGATES Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function gets the potential new owners of a task. The actual owner is excluded from the list.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_APPROVAL.GET_TASK_DELEGATES (
    p_task_id IN NUMBER )
RETURN apex_t_temp_lov_data pipelined;
```

Parameters

Parameter	Description
p_task_id	The task ID.

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,val from table ( apex_approval.get_task_delegates ( p_task_id =>
1234 ) )
```

9.15 GET_TASK_HISTORY Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function gets the approval log for a task.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_APPROVAL.GET_TASK_HISTORY (
    p_task_id          IN NUMBER,
    p_include_all      IN VARCHAR2 DEFAULT 'N' )
RETURN apex_t_approval_log_table pipelined;
```

Parameters

Parameter	Description
p_task_id	The task ID.
p_include_all	If set to Y, the history of all tasks linked to the task with the given task ID is shown. Since Oracle APEX release 22.2, this includes prior Tasks that have been expired.

Returns

A table of approval log entries (type apex_t_approval_log_table) containing the following columns:

- display_msg varchar2(4000)
- event_creator varchar2(255)
- event_creator_lower varchar2(255)
- event_timestamp timestamp with time zone
- event_type varchar2(255)
- event_type_code varchar2(32)
- new_actual_owner varchar2(255)
- new_actual_owner_lower varchar2(255)
- new_priority number
- new_priority_level varchar2(255)
- new_state varchar2(255)
- new_state_code varchar2(32)
- old_actual_owner varchar2(255)
- old_actual_owner_lower varchar2(255)
- old_priority number
- old_priority_level varchar2(255)
- old_state varchar2(255)
- old_state_code varchar2(32)
- outcome varchar2(255)
- outcome_code varchar2(32)

Example

```
select *  from table ( apex_approval.get_task_history ( p_task_id => 1234,
p_include_all => 'Y' ) )
```

9.16 GET_TASK_PARAMETER_OLD_VALUE Function

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function retrieves the old value of a parameter of this task that was updated in the current session. Raises a "No Data Found" error if the parameter does not exist and `p_raise_error` flag is set to TRUE.

Syntax

```
APEX_APPROVAL.GET_TASK_PARAMETER_OLD_VALUE (
    p_task_id          IN NUMBER,
    p_param_static_id  IN VARCHAR2,
    p_raise_error       IN BOOLEAN DEFAULT TRUE )
```

Parameters

Parameter	Description
<code>p_task_id</code>	The Task ID.
<code>p_param_static_id</code>	The static ID of the parameter.
<code>p_raise_error</code>	If TRUE, raises an error if the parameter is not found.

Returns

VARCHAR2 - The old value of this parameter in VARCHAR2 format.

Example

```
BEGIN
    return apex_approval.get_task_parameter_old_value(
        p_task_id          => 1234,
        p_param_static_id  => 'REQ_AMOUNT',
        p_raise_error       => false);
END;
```

9.17 GET_TASK_PARAMETER_VALUE Function

Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function gets the value of a Task parameter. This function can be used in SQL or PL/SQL to get the value of a Task parameter for a given task.

Syntax

```
APEX_APPROVAL.GET_TASK_PARAMETER_VALUE (
    p_task_id          IN NUMBER,
    p_param_static_id IN VARCHAR2,
    p_ignore_not_found IN BOOLEAN DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_param_static_id	The static id of the parameter.
p_ignore_not_found	If set to false (default) and no data is found, a no_data_found exception will be raised. If set to true and no data is found, null will be returned.

Returns

The task parameter value for the given static ID or null.

Exception

no_data_found - In the case where p_ignore_not_found is set to false and no data is found (for example, if the parameter of given name does not exist).

Example

```
DECLARE
    l_req_item varchar2(100);
BEGIN
    l_req_item := apex_approval.get_task_parameter_value(
        p_task_id      => 1234,
        p_param_static_id => 'REQ_ITEM'
    );
    dbms_output.put_line('Parameter REQ_ITEM of task 1234 has value ' || l_req_item);
END;
```

9.18 GET_TASK_PRIORITIES Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function gets the potential new priorities of a task. The actual priority is excluded from the list.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_APPROVAL.GET_TASK_PRIORITIES (
    p_task_id IN NUMBER )
RETURN apex_t_temp_lov_data pipelined;
```

Parameters

Parameter	Description
p_task_id	The task ID.

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,val from table ( apex_approval.get_task_priorities ( p_task_id =>
1234 ) )
```

9.19 GET_TASKS Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function gets the tasks of a user depending on the given context.

Context can be one of the following:

- MY_TASKS - Returns all tasks where the user calling the function is either the Owner or one of the Potential Owners of the task.
- ADMIN_TASKS - Returns all tasks for which the user calling the function is a Business Administrator.
- INITIATED_BY_ME - Returns all tasks where the user calling the function is the Initiator.
- SINGLE_TASK - Returns the task identified by the P_TASK_ID input parameter.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_APPROVAL.GET_TASKS (
    p_context          IN VARCHAR2 DEFAULT apex_approval.c_context_my_tasks,
    p_user             IN VARCHAR2 DEFAULT apex_application.g_user,
    p_task_id          IN NUMBER   DEFAULT NULL,
    p_application_id  IN NUMBER   DEFAULT NULL,
    p_show_expired_tasks IN VARCHAR2 DEFAULT 'N' )
RETURN apex_t_approval_tasks pipelined;
```

Parameters

Parameter	Description
p_context	The list context. Default is MY_TASKS.
p_user	The user to check for. Default is logged-in user. Requires p_context set to MY_TASKS, ADMIN_TASKS or INITIATED_BY_ME.
p_task_id	Filter for a task ID instead of a user. Default is null. Requires p_context set to SINGLE_TASK.
p_application_id	Filter for an application. Default is null (all applications).
p_show_expired_tasks	If set to Y the tasks returned include tasks which are in Expired state.

Returns

A table of tasks (type apex_t_approval_tasks) containing the following columns:

- actual_owner varchar2(255)
- actual_owner_lower varchar2(255)
- app_id number
- badge_css_classes varchar2(255)
- badge_text varchar2(255)
- created_ago varchar2(255)
- created_ago_hours number
- created_by varchar2(255)
- created_on timestamp with time zone
- details_app_id number
- details_app_name varchar2(255)

- details_link_target varchar2(4000)
- due_code varchar2(32)
- due_in varchar2(255)
- due_in_hours number
- due_on timestamp with time zone
- initiator varchar2(255)
- initiator_can_complete varchar2(1)
- initiator_lower varchar2(255)
- is_completed varchar2(1)
- last_updated_by varchar2(255)
- last_updated_on timestamp with time zone
- outcome varchar2(255)
- outcome_code varchar2(32)
- priority number(1)
- priority_level varchar2(255)
- state varchar2(255)
- state_code varchar2(32)
- subject varchar2(1000)
- task_def_id number
- task_def_name varchar2(255)
- task_def_static_id varchar2(255)
- task_id number
- task_type varchar2(8)

Example

```
select * from table ( apex_approval.get_tasks ( p_context => 'MY_TASKS',
p_show_expired_tasks => 'Y' ) )
```

9.20 HANDLE_TASK_DEADLINES Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure handles Task Deadlines for all Tasks in the current Workspace. A background Job performs this work every hour.

Use this API for testing of Task Expiration Policies and "Before Expire" and "Expire" Task Actions.

Syntax

```
APEX_APPROVAL.HANDLE_TASK_DEADLINES;
```

Parameters

Parameter	Description
none	none

Example

```
BEGIN
    apex_approval.handle_task_deadlines;
END;
```

9.21 HAS_TASK_PARAM_CHANGED Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function checks if the value of this task parameter has been modified in the current session. Returns NULL when the parameter does not exist.

Syntax

```
APEX_APPROVAL.HAS_TASK_PARAM_CHANGED (
    p_task_id          IN NUMBER,
    p_param_static_id IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_param_static_id	The static ID of the parameter.

Example

```
BEGIN
    return apex_approval.has_task_param_changed(
        p_task_id      => 1234,
        p_param_static_id => 'REQ_AMOUNT'
```

```
    );
END;
```

9.22 IS_ALLOWED Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function checks whether the given user is permitted to perform a certain operation on a Task.

Syntax

```
APEX_APPROVAL.IS_ALLOWED (
    p_task_id          IN NUMBER,
    p_operation        IN apex_approval.t_task_operation,
    p_user             IN VARCHAR2 DEFAULT apex_application.g_user,
    p_new_participant IN VARCHAR2 DEFAULT NULL )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_operation	The operation to check (see constants c_task_op_###).
p_user	The user to check for. Default is logged in user.
p_new_participant	(Optional) The new assignee in case of Delegate operation.

Returns

TRUE if the user given by p_user is permitted to perform the operation given by p_operation, FALSE otherwise.

Example

```
DECLARE
    l_is_allowed boolean;
BEGIN
    l_is_allowed := apex_approval.is_allowed(
        p_task_id      => 1234,
        p_operation    => apex_approval.c_task_op_delegate
        p_user         => 'STIGER',
        p_new_participant => 'SMOON'
    );
    IF l_is_allowed THEN
        dbms_output.put_line('STIGER is allowed to delegate the task to SMOON for task 1234');
```

```
    END IF;
END;
```

9.23 IS_BUSINESS_ADMIN Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function checks whether the given user is a business administrator for at least one task definition.

Syntax

```
APEX_APPROVAL.IS_BUSINESS_ADMIN (
    p_user          IN VARCHAR2 DEFAULT apex_application.g_user,
    p_application_id IN NUMBER   DEFAULT NULL )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_user	The user to check for. Default is logged-in user.
p_application_id	The application to check for. Default behavior checks against all applications in the workspace.

Returns

TRUE if the user given by p_user is at least in one task definition configured as participant type BUSINESS_ADMIN, FALSE otherwise.

Example

```
DECLARE
    l_is_business_admin boolean;
BEGIN
    l_is_business_admin := apex_approval.is_business_admin(
        p_user => 'STIGER'
    );
    IF l_is_business_admin THEN
        dbms_output.put_line('STIGER is a Business Administrator');
    END IF;
END;
```

9.24 IS_OF_PARTICIPANT_TYPE Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function checks whether the given user is of a certain participant type for a Task.

Syntax

```
APEX_APPROVAL.IS_OF_PARTICIPANT_TYPE (
    p_task_id          IN NUMBER,
    p_participant_type IN t_task_participant_type
                           DEFAULT c_task_potential_owner,
    p_user             IN VARCHAR2
                           DEFAULT wwv_flow_security.g_user)
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_participant_type	The participant type. Can be set to POTENTIAL_OWNER (default) or BUSINESS_ADMIN.
p_user	The user to check for. Default is logged-in user.

Returns

TRUE if the user given by p_user is a participant of given participant type for a given task, FALSE otherwise.

Example

```
DECLARE
    l_is_potential_owner boolean;
BEGIN
    l_is_potential_owner := apex_approval.is_of_participant_type(
        p_task_id      => 1234,
        p_participant_type => apex_approval.c_task_potential_owner,
        p_user         => 'STIGER'
    );
    IF l_is_potential_owner THEN
        dbms_output.put_line('STIGER is a potential owner for task 1234');
    END IF;
END;
```

9.25 REJECT_TASK Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure rejects the task. Only a potential owner or the actual owner of the task can invoke this procedure.

Moves the state of the Task to `Completed` and sets the outcome of the Task to `Rejected`. This is a convenience procedure and equivalent to calling `complete_task` with outcome `apex_approval.c_task_outcome_rejected`.

Syntax

```
APEX_APPROVAL.REJECT_TASK (
    p_task_id           IN NUMBER,
    p_autoclaim        IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
<code>p_task_id</code>	The Task ID.
<code>p_autoclaim</code>	If Task is in state UNASSIGNED then claim the task implicitly.

State Handling

Pre-State: ASSIGNED|UNASSIGNED (`p_autoclaim=true`)

Post-State: COMPLETED

Example

```
BEGIN
    apex_approval.reject_task(
        p_task_id => 1234
    );
END;
```

See Also:

[COMPLETE_TASK Procedure](#)

9.26 RELEASE_TASK Procedure

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure releases an Assigned task from its current owner and sets the task to Unassigned state. Only the current owner of the task can invoke this procedure.

Syntax

```
APEX_APPROVAL.RELEASE_TASK (
    p_task_id           IN NUMBER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.

State Handling

Pre-State: ASSIGNED

Post-State: UNASSIGNED

Example

```
BEGIN
    apex_approval.release_task(
        p_task_id      => 1234
    );
END;
```

9.27 REMOVE_POTENTIAL_OWNER Procedure

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure removes a potential owner of a task. If the user to be removed is *not* an existing potential owner, the API raises an exception.

Only a Business Administrator for the task can run this procedure.

Syntax

```
APEX_APPROVAL.REMOVE_POTENTIAL_OWNER (
    p_task_id           IN NUMBER,
    p_potential_owner   IN VARCHAR2 );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_potential_owner	The potential owner.

Example

The following example removes user "STIGER" as potential owner for Task ID 1234.

```
BEGIN
    apex_approval.remove_potential_owner(
        p_task_id      => 1234,
        p_potential_owner => 'STIGER'
    );
END;
```

9.28 RENEW_TASK Function

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This function reactivates Expired or Errored Tasks. Tasks that have been transitioned to state EXPIRED or ERRORED can be renewed by a Business Administrator.

When a Business Administrator renews a Task, a new Task is created with the given information from the given Task ID. The renewed task is associated with the Expired/Errored Task so that users can review the origin of the Task. This function returns the ID of the renewed task.

Syntax

```
APEX_APPROVAL.RENEW_TASK (
    p_task_id      IN NUMBER,
    p_priority     IN INTEGER DEFAULT NULL,
    p_due_date     IN timestamp with time zone )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_priority	The priority of the renewed Task.
p_due_date	The due date for the renewed Task.

Returns

This function returns the ID of the renewed task.

State Handling

State of original Task: EXPIRED, ERRORED

State of renewed Task: ASSIGNED, UNASSIGNED

Example

```
BEGIN
    apex_approval.renew_task(
        p_task_id          => 1234,
        p_priority         => apex_approval.c_task_priority_high,
        p_due_date         => sysdate + 10
    );
END;
```

9.29 REQUEST_MORE_INFORMATION Procedure

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure requests more information for a task. The owner of a task can request additional information regarding a Task from the initiator. The task then moves to the Information Requested state and can be acted on by the owner only after the initiator submits the requested information.

Syntax

```
APEX_APPROVAL.REQUEST_MORE_INFORMATION (
    p_task_id           IN NUMBER,
    p_text              IN VARCHAR2 )
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_text	Text describing the information requested.

State Handling

Pre-State: ASSIGNED

Post-State: INFO_REQUESTED

Example

```
BEGIN
    apex_approval.request_more_information(
        p_task_id => 1234,
        p_text      => 'Please provide the flight PNR for your travel'
    );
END;
```

See Also:

[SUBMIT_INFORMATION Procedure](#)

9.30 SET_INITIATOR_CAN_COMPLETE Procedure

Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure updates the initiator_can_complete attribute of a task. The task can **not** be COMPLETED or ERRORED. Only a user who is a business administrator for the task can invoke this procedure.

Syntax

```
APEX_APPROVAL.SET_INITIATOR_CAN_COMPLETE (
    p_task_id          IN NUMBER,
    p_initiator_can_complete IN BOOLEAN )
```

Parameters

Parameter	Description
p_task_id	The task ID.
p_initiator_can_complete	TRUE if the initiator is permitted to also approve or reject the task. Otherwise, FALSE.

Example

```
BEGIN
    apex_approval.set_initiator_can_complete(
        p_task_id          => 1234,
        p_initiator_can_complete => true
    );
END;
```

9.31 SET_TASK_DUE Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure sets the due date of a task and can be invoked by the Business Administrator to update the due date of the task.

This API cannot be invoked for a task that is Expired, Errorred, Completed or Canceled.

The due date needs to be in the future, otherwise an exception is thrown when invoking this API.

Syntax

```
APEX_APPROVAL.SET_TASK_DUE (
    p_task_id           IN NUMBER,
    p_due_date          IN timestamp with time zone )
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_due_date	The new due date of the Task.

Example

```
BEGIN
    apex_approval.set_task_due(
```

```

    p_task_id  => 1234,
    p_due_date => sysdate+20
);
END;

```

9.32 SET_TASK_PARAMETER_VALUES Procedure

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure updates the values of the parameter(s) of this task. This procedure only updates the parameters that are marked as "updatable" in the task definition.

Only a Business Administrator or the owner of the task can run this procedure.

Syntax

```
APEX_APPROVAL.SET_TASK_PARAMETER_VALUES (
    p_task_id          IN NUMBER,
    p_parameters       IN t_task_parameters,
    p_raise_error      IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_parameters	The list of changed parameters.
p_raise_error	Default TRUE. When TRUE, the API raises an exception and cancels updates to the parameters. If FALSE, the API ignores raised exceptions if the list contains one or more incorrect parameter static IDs or parameters that are not marked as updatable in the Task Definition. The API updates the rest of the parameters.

Example

```
BEGIN
    apex_approval.set_task_parameter_values(
        p_task_id          => 1234,
        p_parameters       => apex_approval.t_task_parameters(
            1 => apex_approval.t_task_parameter(static_id => 'REQ_DATE',
                                                string_value =>
                                                sysdate+10),
            3 => apex_approval.t_task_parameter(static_id => 'REQ_AMOUNT',
                                                string_value =>
```

```
l_req_amount));
END;
```

9.33 SET_TASK_PRIORITY Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure sets the priority of a task.

This procedure updates the priority of a task. The task can not be COMPLETED or ERRORED. Only a user who is either a Business Administrator for the task or is the initiator of the task can invoke this procedure.

Syntax

```
APEX_APPROVAL.SET_TASK_PRIORITY (
    p_task_id          IN NUMBER,
    p_priority         IN INTEGER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_priority	The task priority (between 1 and 5, 1 being the highest).

Example

```
BEGIN
    apex_approval.set_task_priority(
        p_task_id  => 1234,
        p_priority  => apex_approval.c_task_priority_highest
    );
END;
```

9.34 SUBMIT_INFORMATION Procedure

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [APEX_HUMAN_TASK](#) instead.

This procedure submits information for a task. The initiator of a task can submit additional information regarding a Task for which information has been requested. For example, a travel approver might need airline details from the initiator. The initiator can submit this information to the travel approver using this API.

Syntax

```
APEX_APPROVAL.SUBMIT_INFORMATION (
    p_task_id           IN NUMBER,
    p_text              IN VARCHAR2 )
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_text	Text containing the information submitted.

Example

```
BEGIN
    apex_approval.submit_information(
        p_task_id => 1234,
        p_text      => 'The flight PNR is PN1234'
    );
END;
```

See Also:

[REQUEST_MORE_INFORMATION Procedure](#)

10

APEX_AUTHENTICATION

The APEX_AUTHENTICATION package provides a public API for authentication plug-in.

- Constants
- CALLBACK Procedure
- CALLBACK2 Procedure
- GET_CALLBACK_URL Function
- GET_LOGIN_USERNAME_COOKIE Function
- IS_AUTHENTICATED Function
- IS_PUBLIC_USER Function
- LOGIN Procedure
- LOGOUT Procedure
- PERSISTENT_AUTH_ENABLED Function
- PERSISTENT_COOKIES_ENABLED Function
- POST_LOGIN Procedure
- REMOVE_CURRENT_PERSISTENT_AUTH Procedure
- REMOVE_PERSISTENT_AUTH Procedure
- SAML_CALLBACK Procedure
- SAML_METADATA Procedure
- SEND_LOGIN_USERNAME_COOKIE Procedure

10.1 Constants

The APEX_AUTHENTICATION package uses the following constants.

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

10.2 CALLBACK Procedure

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

 **Tip:**

The parameters which are marked with "OAuth2" should **not** be used for custom callback URLs. They are only used if this procedure is used for Social Sign-In. These parameters are defined by the OAuth2 spec.

Syntax

```
APEX_AUTHENTICATION_CALLBACK (
    --
    -- Custom callback parameters
    --
    p_session_id      IN NUMBER      DEFAULT NULL,
    p_app_id          IN NUMBER      DEFAULT NULL,
    p_ajax_identifier IN VARCHAR2    DEFAULT NULL,
    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,
    p_x08              IN VARCHAR2    DEFAULT NULL,
    p_x09              IN VARCHAR2    DEFAULT NULL,
    p_x10              IN VARCHAR2    DEFAULT NULL,
    --
    -- OAuth2-related parameters
    --
    state              IN VARCHAR2    DEFAULT NULL,
    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,
    hd                 IN VARCHAR2    DEFAULT NULL,
    scope              IN VARCHAR2    DEFAULT NULL,
    realmID            IN VARCHAR2    DEFAULT NULL )
```

Parameters

Parameters	Description
p_session_id	The Oracle APEX session identifier.
p_app_id	The database application identifier.
p_page_id	Optional page identifier.
p_ajax_identifier	The system generated Ajax identifier. See GET_AJAX_IDENTIFIER Function .
p_x01 through p_x10	Optional parameters that the external login passes to the authentication plugin.
state	OAuth2.
code	OAuth2.
error	OAuth2.
error_description	OAuth2.

Parameters	Description
error_uri	OAuth2.
error_reason	OAuth2.
error_code	OAuth2.
error_message	OAuth2.
authuser	OAuth2.
session_state	OAuth2.
prompt	OAuth2.
hd	OAuth2.
scope	OAuth2.
realmID	OAuth2.

Example 1

In this example, a redirect is performed to an external login page and the callback is passed into APEX, 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 APEX, the callback activates the authentication plugin's ajax code. It can take the value of x01 and fetch the actual user data from the shared table.

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

 **See Also:**

- [GET_CALLBACK_URL Function](#)
- [CALLBACK2 Procedure](#)

10.3 CALLBACK2 Procedure

This procedure is an alternative to [CALLBACK Procedure](#).

Syntax

```
APEX_AUTHENTICATION.CALLBACK2 (
  --
  -- Custom callback parameters
  --
  p_session_id      IN NUMBER    DEFAULT NULL,
  p_app_id          IN NUMBER    DEFAULT NULL,
  p_ajax_identifier IN VARCHAR2  DEFAULT NULL,
  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,
  p_x08              IN VARCHAR2  DEFAULT NULL,
  p_x09              IN VARCHAR2  DEFAULT NULL,
  p_x10              IN VARCHAR2  DEFAULT NULL,
  --
  -- OAuth2-related parameters
  --
  state              IN VARCHAR2  DEFAULT NULL,
  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,
  hd                 IN VARCHAR2  DEFAULT NULL,
  scope              IN VARCHAR2  DEFAULT NULL,
  realmID            IN VARCHAR2  DEFAULT NULL )
```

 **See Also:**

[CALLBACK Procedure](#)

10.4 GET_CALLBACK_URL Function

This function is a plug-in 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 plug-in 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

Parameters	Description
p_x01 through p_x10	Optional parameters that the external login passes to the authentication plugin.
p_callback_name	Optional public name of the callback, defaults to apex_authentication.callback.

 **See Also:**

[CALLBACK Procedure](#)

10.5 GET_LOGIN_USERNAME_COOKIE Function

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

Syntax

```
GET_LOGIN_USERNAME_COOKIE (
    p_cookie_name IN VARCHAR2 DEFAULT C_DEFAULT_USERNAME_COOKIE )
RETURN VARCHAR2;
```

Parameters

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

Example

This example is a part of 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.

```
: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:**

[SEND_LOGIN_USERNAME_COOKIE Procedure](#)

10.6 IS_AUTHENTICATED Function

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

Syntax

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

Parameters

None.

Example

This example uses IS_AUTHENTICATED to emit either the username if the user has already logged in or a notification if the user has not.

```
IF apex_authentication.is_authenticated THEN
    sys.hpt.p(apex_escape.html(:APP_USER) || ', you are known to the system');
ELSE
    sys.hpt.p('Please sign in');
END IF;
```

 See Also:

[IS_PUBLIC_USER Function](#)

10.7 IS_PUBLIC_USER Function

This function checks if the user is not authenticated in the session. Returns FALSE if the user is already logged in or TRUE if the user is not authenticated.

Syntax

```
APEX_AUTHENTICATION.IS_PUBLIC_USER
    RETURN BOOLEAN;
```

Parameters

None.

Example

This example uses IS_PUBLIC_USER to display either a notification if the user has not already logged in or the username if the user has not.

```
IF apex_authentication.is_public_user THEN
    sys.htp.p('Please sign in');
ELSE
    sys.htp.p(apex_escape.html(:APP_USER) || ', you are known to the system');
END IF;
```

10.8 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.
 - If result=true: run post-authentication procedure.
 - If result=true: save username in session table.
 - If result=true: set redirect URL to deep link.
 - If result=false: set redirect URL to current page, with an error message in the `notification_msg` parameter.
3. Log authentication result.
4. Redirect.

Syntax

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

Parameters

Parameters	Description
p_username	The user's name.
p_password	The user's password.
p_uppercase_username	If TRUE then p_username is converted to uppercase.
p_set_persistent_auth	If TRUE then persistent authentication cookie is set. Persistent authentication needs to be enabled on instance level.

Example

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

```
BEGIN
    apex_authentication.login (
        p_username => 'JOE USER',
        p_password => 'mysecret' );
END;
```



See Also:

[POST_LOGIN Procedure](#)

10.9 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 );
```

Parameters

Parameters	Description
p_session_id	The Oracle APEX session identifier of the session to close.
p_app_id	The database application identifier.

Example

This example logs the session out.

```
BEGIN
    apex_authentication.logout (
        p_session_id => :APP_SESSION,
        p_app_id => :APP_ID );
END;
```

10.10 PERSISTENT_AUTH_ENABLED Function

This function returns whether persistent authentication is enabled on instance level.

Syntax

```
APEX_AUTHENTICATION.PERSISTENT_AUTH_ENABLED
    return BOOLEAN;
```

Parameters

None.

Example

The following example uses PERSISTENT_AUTH_ENABLED to show a notification.

```
begin
    if apex_authentication.persistent_auth_enabled then
        sys.htp.p('Persistent Authentication enabled');
    else
        sys.htp.p('Persistent Authentication disabled');
    end if;
end;
```

10.11 PERSISTENT_COOKIES_ENABLED Function

This function returns whether persistent cookies are enabled on the instance. Instance administrators can control this value with the parameter WORKSPACE_NAME_USER_COOKIE.

Syntax

```
FUNCTION PERSISTENT_COOKIES_ENABLED
    RETURN BOOLEAN;
```

Returns

- TRUE: WORKSPACE_NAME_USER_COOKIE is set to Y or not set.
- FALSE: WORKSPACE_NAME_USER_COOKIE is set to N.

10.12 POST_LOGIN Procedure

This procedure authenticates the user in the current session. It runs a subset of APEX_AUTHENTICATION.LOGIN, without steps 1 and 2. For steps, see [LOGIN Procedure](#). This procedure is useful in authentication schemes where user credentials checking is performed externally to Oracle APEX.

Syntax

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

Parameters

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

Example

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

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

**See Also:**

[LOGIN Procedure](#)

10.13 REMOVE_CURRENT_PERSISTENT_AUTH Procedure

This procedure removes all Persistent Authentication entries for the user's current browser.

Syntax

```
APEX_AUTHENTICATION.REMOVE_CURRENT_PERSISTENT_AUTH;
```

Parameters

None.

Example

This example invalidates the user's persistent authentication cookies for the current browser and application.

```
apex_authentication.remove_current_persistent_auth;
```

 **See Also:**

[LOGIN Procedure](#)

10.14 REMOVE_PERSISTENT_AUTH Procedure

This procedure removes all Persistent Authentication entries for a user and ends all related sessions in the current workspace.

Syntax

```
APEX_AUTHENTICATION.REMOVE_PERSISTENT_AUTH (
    p_username      IN VARCHAR2 )
```

Parameters

Parameter	Description
p_username	The user's name. If enabled, this procedure only invalidates persistent authentication cookies of this user. If set to NULL, then invalidates all persistent authentication cookies of all users for this workspace.

Example

This example deletes all Persistent Authentication entries for the current user and ends all sessions of this user in the current workspace.

```
apex_authentication.remove_persistent_auth(
    p_username      => :APP_USER );
```

 **See Also:**

[LOGIN Procedure](#)

10.15 SAML_CALLBACK Procedure

Landing resource for SAML authentication. To be called directly from the browser by the SAML identity provider.

Syntax

```
APEX_AUTHENTICATION.SAML_CALLBACK (
    SAMLResponse      IN VARCHAR2 DEFAULT NULL,
    SAMLRequest       IN VARCHAR2 DEFAULT NULL,
    RelayState        IN VARCHAR2 DEFAULT NULL,
    SigAlg            IN VARCHAR2 DEFAULT NULL,
    Signature         IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
SAMLResponse	The base64-encoded SAML response. For GET requests, Oracle APEX assumes that the data is also deflated.
SAMLRequest	Request from the IP to APEX (such as logout). Same format as SAMLRESPONSE.
RelayState	APEX session specific data.
SigAlg	Signature algorithm.
Signature	Signature value.

10.16 SAML_METADATA Procedure

This procedure emits the SAML metadata for the given application or for the Oracle APEX instance.

Syntax

```
APEX_AUTHENTICATION.SAML_METADATA (
    p_app_id      IN NUMBER    DEFAULT NULL )
```

Parameters

Parameter	Description
p_app_id	The ID of the application for which service provider metadata should be generated. If NULL or if the application's SAML authentication is configured to use instance mode, generate metadata using the SAML instance attributes.

Example

The following example downloads SAML metadata for app 101.

```
$ curl https://www.example.com/apex/apex_authentication.saml_metadata?
p_app_id=101
```

10.17 SEND_LOGIN_USERNAME_COOKIE Procedure

This procedure sends a cookie with the username.

Syntax

```
APEX_AUTHENTICATION.SEND_LOGIN_USERNAME_COOKIE (
    p_username      IN VARCHAR2,
    p_cookie_name   IN VARCHAR2 DEFAULT c_default_username_cookie,
    p_consent       IN BOOLEAN  DEFAULT FALSE )
```

Parameters

Parameters	Description
p_username	The user's name.
p_cookie_name	The cookie name which stores p_username in the browser.
p_consent	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. This parameter is ignored and no cookie gets sent if PERSISTENT_COOKIES_ENABLED returns false.

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.

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

See Also:

- [GET_LOGIN_USERNAME_COOKIE Function](#)
- [PERSISTENT_COOKIES_ENABLED Function](#)

11

APEX_AUTHORIZATION

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](#)

11.1 ENABLE_DYNAMIC_GROUPS Procedure

This procedure enables groups in the current session. These groups do not have to be created in the Oracle APEX workspace repository, but can be loaded from an LDAP repository or retrieved from a trusted HTTP header. 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 immediately after authentication, for example, in a post-authentication procedure.

Syntax

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

Parameters

Parameter	Description
p_group_names	Table of group names.

Example

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

```
BEGIN
    apex_authorization.enable_dynamic_groups (
        p_group_names => apex_t_varchar2('SALES', 'HR') );
END;
```

 See Also:

[View APEX_WORKSPACE_SESSION_GROUPS](#) and [View APEX_WORKSPACE_GROUP_GROUPS](#)

11.2 IS_AUTHORIZED Function

This function determines 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 APEX App Builder User's Guide*

Syntax

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

Parameters

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

Returns

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

Example

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

```
BEGIN
    sys.htp.p('User Is Admin: ' ||
        case apex_authorization.is_authorized (
            p_authorization_name => 'User Is Admin' )
        WHEN true THEN 'YES'
        WHEN false THEN 'NO'
        ELSE 'null'
        END);
END;
```

11.3 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;
```

12

APEX_AUTOMATION

The `APEX_AUTOMATION` package provides automated functionality to your environment. Automations are a sequential set of actions which are triggered by query results. Use automations to monitor data and then perform the appropriate action, such as auto-approving specific requests and sending email alerts.

- [ABORT Procedure \(Deprecated\)](#)
- [DISABLE Procedure](#)
- [ENABLE Procedure](#)
- [EXECUTE Procedure Signature 1](#)
- [EXECUTE Procedure Signature 2](#)
- [EXECUTE for Query Context Procedure](#)
- [EXIT Procedure](#)
- [GET_LAST_RUN Function](#)
- [GET_LAST_RUN_TIMESTAMP Function](#)
- [GET_SCHEDULER_JOB_NAME Function](#)
- [IS_RUNNING Function](#)
- [LOG_ERROR Procedure](#)
- [LOG_INFO Procedure](#)
- [LOG_WARN Procedure](#)
- [RESCHEDULE Procedure](#)
- [SKIP_CURRENT_ROW Procedure](#)
- [TERMINATE Procedure](#)

12.1 ABORT Procedure (Deprecated)

! **Important:**

This API is deprecated and will be removed in a future release.

Use `APEX_AUTOMATION.TERMINATE` instead.

This procedure terminates a currently executing automation.

Syntax

```
APEX_AUTOMATION.ABORT (
    p_application_id      IN NUMBER    DEFAULT {current application id},
    p_static_id           IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to terminate.

Example

The following example aborts the currently executing automation `my_emp_table_automation` in application 152. If the automation is not running, nothing happens.

```
BEGIN
    apex_automation.abort(
        p_application_id => 152,
        p_static_id      => 'my_emp_table_automation' );
END;
```

 **See Also:**

- [TERMINATE Procedure](#)

12.2 DISABLE Procedure

This procedure stops the automation from executing automatically.

Syntax

```
APEX_AUTOMATION.DISABLE (
    p_application_id      IN NUMBER    DEFAULT {current application id},
    p_static_id           IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to disable.

Examples

This example disables the automation `my_emp_table_automation` in application 152.

```
BEGIN
    apex_automation.disable(
        p_application_id  => 152,
        p_static_id       => 'my_emp_table_automation' );
END;
```

12.3 ENABLE Procedure

This procedure enables the automation for normal execution.

Syntax

```
APEX_AUTOMATION.ENABLE (
    p_application_id      IN NUMBER      DEFAULT {current application id},
    p_static_id           IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to enable.

Examples

This example enables the automation `my_emp_table_automation` in application 152.

```
BEGIN
    apex_automation.enable(
        p_application_id  => 152,
        p_static_id       => 'my_emp_table_automation' );
END;
```

12.4 EXECUTE Procedure Signature 1

This procedure executes an automation.

Syntax

```
APEX_AUTOMATION.EXECUTE (
    p_application_id      IN NUMBER          DEFAULT {current
application id},
    p_static_id           IN VARCHAR2,
    p_filters              IN apex_exec.t_filters      DEFAULT
apex_exec.c_empty_filters,
```

```
p_order_bys      IN apex_exec.t_order_bys      DEFAULT
apex_exec.c_empty_order_bys )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to execute.
p_filters	Additional filters to apply to the automation query.
p_order_bys	ORDER BY clauses to apply to the automation query.

Example

This example executes the automation `my_emp_table_automation` and applies a filter to the automation query on the `DEPTNO` column (`DEPTNO = 10`).

```
DECLARE
    l_filters apex_exec.t_filters;
BEGIN
    apex_session.create_session( 100, 1, 'ADMIN' );

    apex_exec.add_filter(
        p_filters      => l_filters,
        p_column_name  => 'DEPTNO',
        p_filter_type   => apex_exec.c_filter_eq,
        p_value         => 10 );

    apex_automation.execute(
        p_static_id     => 'my_emp_table_automation',
        p_filters        => l_filters );
END;
```

12.5 EXECUTE Procedure Signature 2

This procedure executes an automation.

Syntax

```
APEX_AUTOMATION.EXECUTE (
    p_application_id    IN NUMBER DEFAULT {current application id},
    p_static_id         IN VARCHAR2,
    p_run_in_background IN BOOLEAN )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to execute.

Parameter	Description
p_run_in_background	If TRUE, synchronization runs in the background as a one-time DBMS_SCHEDULER job.

Example

This example executes the automation `my_emp_table_automation` in the background.

```
BEGIN
    apex_session.create_session( 100, 1, 'ADMIN' );

    apex_automation.execute(
        p_static_id      => 'my_emp_table_automation',
        p_run_in_background => true );
END;
```

12.6 EXECUTE for Query Context Procedure

This procedure executes automation actions for a given query context. The columns returned by the query context match those defined in the automation query, especially when columns are referenced as bind variables in the actions code.

Syntax

```
APEX_AUTOMATION.EXECUTE (
    p_application_id     IN NUMBER    DEFAULT {current application id},
    p_static_id          IN VARCHAR2,
    p_query_context      IN apex_exec.t_context )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to execute.
p_query_context	The context to run the actions for the query.

Examples

This example executes the actions defined in the automation `my_emp_table_automation`, but uses a different query context.

```
DECLARE
    l_context apex_exec.t_context;
BEGIN
    apex_session.create_session( 100, 1, 'ADMIN' );

    l_context := apex_exec.open_query_context(
                    p_location      => apex_exec.c_location_local_db,
                    p_sql_query     => 'select * from emp_copy_table' );
```

```
apex_automation.execute(
    p_static_id      => 'my_emp_table_automation',
    p_query_context  => l_context );
END;
```

12.7 EXIT Procedure

This procedure exits automation processing, including for remaining rows. Use this procedure in automation action code.

Syntax

```
APEX_AUTOMATION.EXIT (
    p_log_message    IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_log_message	Message to write to the automation log.

Examples

This example cancels the automation if a salary higher than 10,000 is found. The automation uses `select * from emp` as the automation query.

```
BEGIN
    IF :SQL > 10000 THEN
        apex_automation.exit( p_log_message => 'Dubious SAL value found. Exit
automation.' );
    ELSE
        my_logic_package.process_emp(
            p_empno  => :EMPNO,
            p_sal    => :SAL,
            p_deptno => :DEPTNO );
    END IF;
END;
```

12.8 GET_LAST_RUN Function

This function returns the last run of the automation as a `TIMESTAMP WITH TIME ZONE` type. Use this function within automation action code or the automation query.

Syntax

```
APEX_AUTOMATION.GET_LAST_RUN
    RETURN timestamp with time zone;
```

Returns

Return	Description
*	Timestamp of the previous automation run.

Examples

This example automation only selects rows from a table which have the CREATED_AT column after the last run of the automation.

```
select *
  from {table}
 where created_at > apex_automation.get_last_run;
```

12.9 GET_LAST_RUN_TIMESTAMP Function

This function retrieves information about the latest automation run.

Syntax

```
APEX_AUTOMATION.GET_LAST_RUN_TIMESTAMP (
  p_application_id      IN NUMBER    DEFAULT {current application id},
  p_static_id           IN VARCHAR2 )
RETURN timestamp with time zone;
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to execute.

Returns

Return	Description
*	Timestamp of the last successful automation run.

Examples

This example retrieves the timestamp of the last successful run of the my_emp_table_automation.

```
DECLARE
  l_last_run_ts timestamp with time zone;
BEGIN
  apex_session.create_session( 100, 1, 'ADMIN' );
  l_last_run := apex_automation.get_last_run_timestamp(
    p_static_id    => 'my_emp_table_automation' );

  dbms_output.put_line( 'The automation''s last run was as of: ' ||
```

```

l_last_run );
END;

```

12.10 GET_SCHEDULER_JOB_NAME Function

This procedure returns the name which is used for the scheduler job when the automation executes.

Syntax

```

APEX_AUTOMATION.GET_SCHEDULER_JOB_NAME (
    p_application_id      IN NUMBER      DEFAULT {current application id},
    p_static_id           IN VARCHAR2 )
    RETURN VARCHAR2;

```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation.

Returns

The name of the the scheduler job which is generated to execute this automation.

Example

The following example returns the name of the scheduler job which executes the automation with the static ID my_emp_table_automation.

```

BEGIN
    dbms_output.put_line(
        apex_automation.get_scheduler_job_name(
            p_application_id => 152,
            p_static_id      => 'my_emp_table_automation' ) );
    -- ==> APEX$AUTOMATION_2167837869128719
END;

```

12.11 IS_RUNNING Function

This function determines whether a given automation is currently running.

Syntax

```

APEX_AUTOMATION.IS_RUNNING (
    p_application_id      IN NUMBER      DEFAULT {current application id},
    p_static_id           IN VARCHAR2 )
    RETURN BOOLEAN;

```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation.

Returns

If TRUE, the automation is currently running.

Example

The following example prints out whether the automation is currently running.

```
BEGIN
    IF apex_automation.is_running(p_application_id => 152, p_static_id =>
'my_emp_table_automation') THEN
        dbms_output.put_line( 'The Automation is currently running.' );
    ELSE
        dbms_output.put_line( 'The Automation is currently not running.' );
    END IF;
END;
```

12.12 LOG_ERROR Procedure

This procedure writes a log entry with severity ERROR and is to be used within automation code.

Syntax

```
APEX_AUTOMATION.LOG_ERROR (
    p_message    IN VARCHAR2 )
```

Parameters

Parameter	Description
p_message	Message to write to the automation log.

Example

This example writes some log information. The automation uses select * from emp as the automation query.

```
BEGIN
    IF :SAL > 10000 THEN
        apex_automation.log_error(
            p_message => 'High Salary found for empno: ' || :EMPNO );
    ELSE
        my_logic_package.process_emp(
            p_empno => :EMPNO,
            p_sal => :SAL,
            p_depto => :DEPTNO );
```

```
    END IF;
END;
```

12.13 LOG_INFO Procedure

This procedure writes a log entry with severity of `INFO` which can be used within automation code.

Syntax

```
APEX_AUTOMATION.LOG_INFO (
    p_message    IN VARCHAR2 )
```

Parameters

Parameter	Description
<code>p_message</code>	Message to write to the automation log.

Example

This example writes some log information. The automation uses `select * from emp` as the automation query.

```
BEGIN
    IF :SAL > 10000 THEN
        apex_automation.log_info( p_message => 'High Salary found for empno:
' || :EMPNO );
    END IF;
    my_logic_package.process_emp(
        p_empno  => :EMPNO,
        p_sal     => :SAL,
        p_deptno  => :DEPTNO );
END;
```

12.14 LOG_WARN Procedure

This procedure writes a log entry with severity `WARN` and is to be used within automation code.

Syntax

```
APEX_AUTOMATION.LOG_WARN (
    p_message    IN VARCHAR2 )
```

Parameters

Parameter	Description
<code>p_message</code>	Message to write to the automation log.

Examples

This example writes some log information. The automation uses select * from emp as the automation query.

```
BEGIN
    IF :SAL > 10000 THEN
        apex_automation.log_warn(
            p_message => 'High Salary found for empno: ' || :EMPNO );
    END IF;
    my_logic_package.process_emp(
        p_empno => :EMPNO,
        p_sal    => :SAL,
        p_deptno => :DEPTNO );
END;
```

12.15 RESCHEDULE Procedure

This procedure sets the next scheduled execution date of a "polling" automation to now so that the main automation execution job executes the automation as soon as possible. If the automation is currently running, it will not restart.

Syntax

```
APEX_AUTOMATION.RESCHEDULE (
    p_application_id    IN NUMBER                               DEFAULT {current
application id},
    p_static_id         IN VARCHAR2,
    p_next_run_at       IN timestamp with time zone      DEFAULT systimestamp )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to execute.
p_next_run_at	Timestamp of the next automation run.

Examples

This example sets the automation `my_emp_table_automation` to execute in the background right now.

```
BEGIN
    apex_session.create_session( 100, 1, 'ADMIN' );

    apex_automation.reschedule(
        p_static_id  => 'my_emp_table_automation' );
END;
```

12.16 SKIP_CURRENT_ROW Procedure

This procedure skips processing of the current row and continues with the next one. Use this procedure in automation action code.

Syntax

```
APEX_AUTOMATION.SKIP_CURRENT_ROW (
    p_log_message    IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_log_message	Message to write to the automation log.

Examples

This example skips the rest of processing for the current row (PRESIDENT row). The automation uses select * from emp as the automation query.

```
BEGIN
    IF :ENAME = 'PRESIDENT' THEN
        apex_automation.skip_current_row( p_log_message => 'PRESIDENT
skipped' );
    ELSE
        my_logic_package.process_emp(
            p_empno  => :EMPNO,
            p_sal    => :SAL,
            p_deptno => :DEPTNO );
    END IF;
END;
```

12.17 TERMINATE Procedure

This procedure terminates a currently executing automation.

Syntax

```
APEX_AUTOMATION.TERMINATE (
    p_application_id    IN NUMBER    DEFAULT {current application id},
    p_static_id         IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_static_id	Static ID of the automation to terminate.

Example

The following example terminates the currently executing automation `my_emp_table_automation` in application 152. If the automation is not running, nothing happens.

```
BEGIN
    apex_automation.terminate(
        p_application_id => 152,
        p_static_id      => 'my_emp_table_automation' );
END;
```

13

APEX_BACKGROUND_PROCESS

This package enables background process reporting (status and progress) and the option to forcefully cancel a running process.

- [Constants](#)
- [Data Types](#)
- [ABORT Procedure Signature 1 \(Deprecated\)](#)
- [ABORT Procedure Signature 2 \(Deprecated\)](#)
- [GET_CURRENT_EXECUTION Function](#)
- [GET_EXECUTION Function](#)
- [SET_PROGRESS Procedure](#)
- [SET_STATUS Procedure](#)
- [TERMINATE Procedure Signature 1](#)
- [TERMINATE Procedure Signature 2](#)

13.1 Constants

The APEX_BACKGROUND_PROCESS package uses the following constants.

```
-- subtype t_execution_state is varchar2(9);
--
-- An execution was submitted, but the coordinator job has not picked it up
-- for execution yet.
--
c_status_enqueued    constant t_execution_state := 'ENQUEUED';
--
-- The coordinator job picked up the execution and started an executor job
-- using the database scheduler, but the scheduler did not start this job yet.
--
c_status_scheduled   constant t_execution_state := 'SCHEDULED';
--
-- The executor job for this background execution is currently executing.
--
c_status_executing   constant t_execution_state := 'EXECUTING';
--
-- The execution finished successfully.
--
c_status_success     constant t_execution_state := 'SUCCESS';
--
-- An unhandled error arose during execution.
--
c_status_failed      constant t_execution_state := 'FAILED';
--
-- The execution was terminated.
```

```
--  
c_status_terminated  constant t_execution_state := 'ABORTED';  
-- Deprecated:  
c_status_aborted      constant t_execution_state := 'ABORTED';
```

13.2 Data Types

The APEX_BACKGROUND_PROCESS package uses the following data types.

Record describing an execution running in the background

```
type t_execution is record (  
    id                      NUMBER,  
    state                   t_execution_state,  
    --  
    current_exec_process_id NUMBER,  
    --  
    last_status_message     VARCHAR2(32767),  
   sofar                   NUMBER,  
    totalwork               NUMBER );
```

Attributes

Attribute	Description
execution_id	ID of the execution.
state	State of the execution, see <code>t_execution_state</code> constants.
current_exec_process_id	ID of the currently executing child process.
context_value	Context value passed when the execution was submitted.
last_status_message	Last status message set by the developer.
sofar	Units of work already processed by the page process.
totalwork	Total units of work to process by the page process.

See Also:

- [Constants](#)

13.3 ABORT Procedure Signature 1 (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use APEX_BACKGROUND_PROCESS.TERMINATE instead.

This procedure aborts all executions of an execution chain.

Syntax

```
APEX_BACKGROUND_PROCESS.ABORT (
    p_application_id      IN NUMBER DEFAULT apex_application.g_flow_id,
    p_process_id          IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	ID of the application containing the process.
p_process_id	ID of the execution chain to abort executions for.

Example

The following example aborts all executions for process 9023498034890234890.

```
BEGIN
    apex_background_process.abort(
        p_application_id => 100,
        p_process_id     => 9023498034890234890 );
END;
```

13.4 ABORT Procedure Signature 2 (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use APEX_BACKGROUND_PROCESS.TERMINATE instead.

This procedure aborts a specific execution of an execution chain.

Syntax

```
APEX_BACKGROUND_PROCESS.ABORT (
    p_application_id      IN NUMBER DEFAULT apex_application.g_flow_id,
    p_execution_id        IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	ID of the application containing the process.
p_execution_id	ID of the execution to abort.

Example

The following example aborts background execution 4711.

```
BEGIN
    apex_background_process.abort(
        p_application_id => 100,
        p_execution_id   => 4711 );
END;
```

13.5 GET_CURRENT_EXECUTION Function

This function returns the status of the current execution. This function is called from within the background process to get its own execution ID.

If the function is not called from a page process running in the background, an empty record is returned.

Syntax

```
APEX_BACKGROUND_PROCESS.GET_CURRENT_EXECUTION
    RETURN t_execution;
```

Parameters

None.

Returns

T_EXECUTION record with status information for the current execution.

Example

The following example retrieves Status information of the currently running background execution.

```
DECLARE
    l_execution apex_background_process.t_execution;
BEGIN
    l_execution := apex_background_process.get_current_execution;
```

```

    sys.dbms_output.put_line( 'Execution ID: ' || l_execution.id );
END;

=> Execution ID: 4711

```

13.6 GET_EXECUTION Function

This function returns the current status of a specific execution ID.

Syntax

```

APEX_BACKGROUND_PROCESS.GET_EXECUTION (
    p_application_id      IN NUMBER DEFAULT apex_application.g_flow_id,
    p_execution_id        IN NUMBER )
RETURN t_execution;

```

Parameters

Parameter	Description
p_application_id	ID of the application containing the process.
p_execution_id	ID of the execution to get status for.

Returns

This function returns `t_execution` record with current status information for this execution.

Example

The following example retrieves Status information for execution ID 4711.

```

DECLARE
    l_execution apex_background_process.t_execution;
BEGIN
    l_execution := apex_background_process.get_execution(
        p_application_id      => 100,
        p_execution_id        => 4711 );

    sys.dbms_output.put_line( 'Execution State: ' || l_execution.state );
END;

=> Execution State: EXECUTING

```

13.7 SET_PROGRESS Procedure

This procedure sets progress of an execution. This procedure must be called from within PL/SQL code.

Use the `GET_EXECUTION` function to retrieve information.

Syntax

```
APEX_BACKGROUND_PROCESS.SET_PROGRESS (
    p_totalwork IN NUMBER DEFAULT NULL,
    p_sofar      IN NUMBER )
```

Parameters

Parameter	Description
p_totalwork	Total units of work to be processed by the background process.
p_sofar	Units of work being processed so far.

Example 1

The following example demonstrates a PL/SQL page process running in the background with a known total amount of work to process. Progress is reported to the Oracle APEX engine as follows.

```
BEGIN
    for i in 1 .. 1000 loop
        do_something( p_param => i );
        apex_background_process.set_progress(
            p_totalwork    => 1000,
            p_sofar        => i );
    END loop;
END;
```

Example 2

The following example demonstrates a PL/SQL page process running in the background with an unknown total amount work to process. Progress is reported to the APEX engine as follows.

```
DECLARE
    l_rows_processed pls_integer := 1;
BEGIN
    for i ( select * from emp ) loop
        do_something( p_param => i.empno );
        apex_background_process.set_progress(
            p_sofar        => l_rows_processed );
        l_rows_processed := l_rows_processed + 1;
    END loop;
END;
```

See Also:

- [GET_EXECUTION Function](#)

13.8 SET_STATUS Procedure

This procedure sets status for an execution chain. This procedure must be called from within PL/SQL code.

Use the `GET_EXECUTION` function to retrieve status messages.

Syntax

```
APEX_BACKGROUND_PROCESS.SET_STATUS (
    p_message    IN VARCHAR2 );
```

Parameters

Parameter	Description
<code>p_message</code>	Current status message for the page chain.

Example

The following example demonstrates a PL/SQL page process running in the background. After each unit of work, a status message is being reported to the APEX engine.

```
DECLARE
    l_result varchar2(255);
BEGIN
    apex_background_process.set_status( 'Part A: Process Orders' );
    for i in ( select *
                from orders
               where status = 'OPEN' )
    LOOP
        l_result := process_order( p_param => i.order_id );
    END LOOP;
    apex_background_process.set_status( 'Part B: Process Bills' );
    for i in ( select *
                from orders
               where status = 'DELIVERED' )
    LOOP
        l_result := emit_bill( p_param => i.order_id );
    END LOOP;
END;
```

See Also:

- [GET_EXECUTION Function](#)

13.9 TERMINATE Procedure Signature 1

This procedure terminates all executions of an execution chain.

Syntax

```
APEX_BACKGROUND_PROCESS.TERMINATE (
    p_application_id      IN NUMBER DEFAULT apex_application.g_flow_id,
    p_process_id          IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	ID of the application containing the process.
p_process_id	ID of the execution chain containing the executions to terminate.

Example

The following example terminates all executions for process 9023498034890234890.

```
BEGIN
    apex_background_process.terminate(
        p_application_id => 100,
        p_process_id     => 9023498034890234890 );
END;
```

13.10 TERMINATE Procedure Signature 2

This procedure terminates a specific execution of an execution chain.

Syntax

```
APEX_BACKGROUND_PROCESS.TERMINATE (
    p_application_id      IN NUMBER DEFAULT apex_application.g_flow_id,
    p_execution_id        IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	ID of the application containing the process.
p_execution_id	ID of the execution to terminate.

Example

The following example aborts background execution 4711.

```
BEGIN
    apex_background_process.terminate(
        p_application_id => 100,
        p_execution_id   => 4711 );
END;
```

14

APEX_BARCODE

The APEX_BARCODE package contains the implementation to generate different types of barcodes. The supported output types are SVG value or PNG file BLOB.

- [GET_CODE128_PNG Function](#)
- [GET_CODE128_SVG Function](#)
- [GET_EAN8_PNG Function](#)
- [GET_EAN8_SVG Function](#)
- [GET_QRCODE_PNG Function](#)
- [GET_QRCODE_SVG Function](#)

14.1 GET_CODE128_PNG Function

This function generates a Code 128 barcode, configured according to the specified options, and returns a BLOB in PNG format.

Syntax

```
APEX_BARCODE.GET_CODE128_PNG (
    p_value          IN VARCHAR2,
    p_scale          IN NUMBER   DEFAULT c_default_scale,
    p_foreground_color IN VARCHAR2 DEFAULT c_default_foreground_color,
    p_background_color IN VARCHAR2 DEFAULT NULL )
RETURN BLOB;
```

Parameters

Parameter	Description
p_value	Value to be encoded into the Code 128 barcode.
p_scale	Makes the original PNG p_scale times larger (integer 1-10). Default 1. The original size is determined by the input length.
p_foreground_color	Foreground color. Must be in hex code. Default #000000.
p_background_color	Background color. Must be in hex code. Default null (transparent).

Returns

The Code 128 barcode PNG image file.

Example

The following example generates a PNG Code 128-type barcode file with specified scale, foreground color, and background color.

```
DECLARE
    l_output blob;
BEGIN
    l_output := apex_barcode.get_code128_png(
        p_value          => 'apex.oracle.com',
        p_scale          => 1,
        p_foreground_color => '#4cd964',
        p_background_color => '#c7c7cc' );
END;
```

14.2 GET_CODE128_SVG Function

This function generates a Code 128 barcode, configured according to the specified options, and returns a CLOB in SVG format.

Syntax

```
APEX_BARCODE.GET_CODE128_SVG (
    p_value          IN VARCHAR2,
    p_size           IN NUMBER   DEFAULT c_default_size,
    p_foreground_color IN VARCHAR2 DEFAULT c_default_foreground_color,
    p_background_color IN VARCHAR2 DEFAULT NULL )
RETURN CLOB;
```

Parameters

Parameter	Description
p_value	Value to be encoded into the Code 128 barcode.
p_size	Size of the Code 128 barcode (in pixels). Default 256px.
p_foreground_color	Foreground color. Must be in hex code. Default #000000.
p_background_color	Background color. Must be in hex code. Default null (transparent).

Returns

The SVG value of the Code 128 barcode.

Example

The following example generates an SVG Code 128-type barcode with specified foreground color and background color.

```
DECLARE
    l_output clob;
BEGIN
    l_output := apex_barcode.get_code128_svg(
```

```

    p_value          => 'apex.oracle.com',
    p_foreground_color => '#4cd964',
    p_background_color => '#c7c7cc' );

  sys.dbms_output.put_line( l_output );

END;

```

14.3 GET_EAN8_PNG Function

This function generates an EAN 8 barcode that is configured according to the specified options, and returns a BLOB in PNG format.

Syntax

```
APEX_BARCODE.GET_EAN8_PNG (
    p_value      IN VARCHAR2,
    p_scale       IN NUMBER  DEFAULT c_default_scale,
    p_foreground_color IN VARCHAR2 DEFAULT c_default_foreground_color,
    p_background_color  IN VARCHAR2 DEFAULT NULL )
RETURN BLOB;
```

Parameters

Parameter	Description
p_value	Value to be encoded into the EAN 8 barcode. Must be numeric with a maximum of 8 characters.
p_scale	Makes the original PNG p_scale times larger (integer 1-10). Default is 1. The original size is determined by the input length.
p_foreground_color	Foreground color. Must be in hex code. Defaults is #000000.
p_background_color	Background color. Must be in hex code. Default is null (transparent).

Returns

The EAN 8 barcode PNG image file.

Raises

WWV_FLOW_BARCODE_API.NUMERIC_INPUT_ERROR: when p_value exceeds 8 characters.

Example

The following example generates a PNG EAN 8 type of barcode file with desired scale, foreground color, and background color.

```

DECLARE
  l_output blob;
BEGIN
  l_output := apex_barcode.get_ean8_png(
    p_value          => '12345678',
    p_scale         => 1,
    p_foreground_color => '#4cd964',

```

```

    p_background_color => '#c7c7cc' );
END;

```

14.4 GET_EAN8_SVG Function

This function generates an EAN 8 barcode that is configured according to the specified options, and returns a CLOB in SVG format.

Syntax

```

APEX_BARCODE.GET_EAN8_SVG (
    p_value          IN VARCHAR2,
    p_size           IN NUMBER   DEFAULT c_default_size,
    p_foreground_color IN VARCHAR2 DEFAULT c_default_foreground_color,
    p_background_color IN VARCHAR2 DEFAULT NULL )
RETURN CLOB;

```

Parameters

Parameter	Description
p_value	Value to be encoded into the EAN 8 Barcode. Format is numeric with a maximum of 8 digits.
p_size	Size of the EAN 8 Barcode (in pixels). Default 256px.
p_foreground_color	Foreground color. Must be in hex code. Default #000000.
p_background_color	Background color. Must be in hex code. Default null (transparent).

Returns

The SVG value of the EAN 8 barcode.

Raises

WWV_FLOW_BARCODE_API.NUMERIC_INPUT_ERROR: when p_value exceeds 8 digits.

Example

The following example generates an SVG EAN 8 type of barcode with specified foreground and background colors.

```

DECLARE
    l_output clob;
BEGIN
    l_output := apex_barcode.get_ean8_svg(
        p_value          => '12345678',
        p_foreground_color => '#4cd964',
        p_background_color => '#c7c7cc');
    sys.dbms_output.put_line( l_output );
END;

```

14.5 GET_QRCODE_PNG Function

This function generates a QR code that is configured according to the specified options and returns a BLOB in PNG format.

Syntax

```
APEX_BARCODE.GET_QRCODE_PNG (
    p_value          IN VARCHAR2,
    p_scale          IN NUMBER      DEFAULT c_default_scale,
    p_quiet          IN NUMBER      DEFAULT c_default_quiet,
    p_eclevel         IN t_eclevel_type DEFAULT c_default_eclevel,
    p_foreground_color IN VARCHAR2   DEFAULT c_default_foreground_color,
    p_background_color IN VARCHAR2   DEFAULT NULL )
RETURN BLOB;
```

Parameters

Parameter	Description
p_value	Value to be encoded into the QR Code.
p_scale	Makes the original PNG p_scale times larger (integer 1-10). Default 1. The original size is determined by the input length.
p_quiet	Blank area (positive integer value) around the QR Code used to help the scanners clearly distinguish the QR Code from its surroundings for good scannability. Defaults 1.
p_eclevel	The error-correction level. The level determines the percentage of the total QR code that can be dirty or damaged and still be valid. Default c_eclevel_type_high. Possible values: <ul style="list-style-type: none"> • c_eclevel_type_low - 7% of data bytes can be restored. • c_eclevel_type_medium - 15% of data bytes can be restored. • c_eclevel_type_quartile - 25% of data bytes can be restored. • c_eclevel_type_high - 30% of data bytes can be restored.
p_foreground_color	Foreground color. Must be in hex format. Default #000000.
p_background_color	Background color. Must be in hex format. Default null (transparent).

Returns

The QR code PNG image file.

Example

The following example generates a QR code PNG file with a determined foreground and background color. This function is usually used when a QR code image file is needed.

```
DECLARE
    l_output blob;
BEGIN
    l_output := apex_barcode.get_qrcode_png(
        p_value          => 'apex.example.com',
        p_scale          => 1,
        p_quiet          => 5,
        p_eclevel        => c_eclevel_type_high,
        p_foreground_color => '#4cd964',
        p_background_color => '#c7c7cc' );
END;
```

14.6 GET_QRCODE_SVG Function

This function generates a QR code that is configured according to the specified options and returns a CLOB in SVG format.

Syntax

```
APEX_BARCODE.GET_QRCODE_SVG (
    p_value          IN VARCHAR2,
    p_size           IN NUMBER      DEFAULT c_default_size,
    p_quiet          IN NUMBER      DEFAULT c_default_quiet,
    p_eclevel        IN t_eclevel_type DEFAULT c_default_eclevel,
    p_foreground_color IN VARCHAR2   DEFAULT c_default_foreground_color,
    p_background_color IN VARCHAR2   DEFAULT NULL )
RETURN CLOB;
```

Parameters

Parameter	Description
p_value	Value to be encoded into the QR code.
p_size	Size of the QR code (in pixels). Defaults to 256px.
p_foreground_color	Foreground color. Must be in hex format. Default #000000.
p_background_color	Background color. Must be in hex format. Default null (transparent).
p_quiet	Blank area (positive integer value) around the QR Code used to help the scanners clearly distinguish the QR code from its surroundings for good scannability. Defaults to 1.

Parameter	Description
p_eclevel	<p>The error-correction level. The level determines the percentage of the total QR code that can be dirty or damaged and still be valid.</p> <p>Default c_eclevel_type_high.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • c_eclevel_type_low - 7% of data bytes can be restored. • c_eclevel_type_medium - 15% of data bytes can be restored. • c_eclevel_type_quartile - 25% of data bytes can be restored. • c_eclevel_type_high - 30% of data bytes can be restored.

Returns

The SVG value of the QR code.

Example

Generates an SVG QR code with a determined foreground and background color. This function is usually used in rendering QR code page item.

```

DECLARE
    l_output clob;
BEGIN
    l_output := apex_barcode.get_qrcode_svg(
        p_value          => 'apex.oracle.com',
        p_foreground_color => '#4cd964',
        p_background_color => '#c7c7cc' );
    sys.dbms_output.put_line( l_output );
END;

```

15

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](#)
- [ADD_MEMBER Procedure](#)
- [ADD_MEMBER Function](#)
- [ADD_MEMBERS Procedure](#)
- [COLLECTION_EXISTS Function](#)
- [COLLECTION_HAS_CHANGED Function](#)
- [COLLECTION_MEMBER_COUNT Function](#)
- [CREATE_COLLECTION Procedure](#)
- [CREATE_COLLECTION_FROM_QUERY Procedure](#)
- [CREATE_COLLECTION_FROM_QUERY2 Procedure](#)
- [CREATE_COLLECTION_FROM_QUERY_B Procedure](#)
- [CREATE_COLLECTION_FROM_QUERY_B Procedure \(No bind version\)](#)
- [CREATE_COLLECTION_FROM_QUERYB2 Procedure](#)
- [CREATE_COLLECTION_FROM_QUERYB2 Procedure \(No bind version\)](#)
- [CREATE_OR_TRUNCATE_COLLECTION Procedure](#)
- [DELETE_ALL_COLLECTIONS Procedure](#)
- [DELETE_ALL_COLLECTIONS_SESSION Procedure](#)
- [DELETE_COLLECTION Procedure](#)
- [DELETE_MEMBER Procedure](#)
- [DELETE_MEMBERS Procedure](#)
- [GET_MEMBER_MD5 Function](#)
- [MERGE_MEMBERS Procedure](#)
- [MOVE_MEMBER_DOWN Procedure](#)
- [MOVE_MEMBER_UP Procedure](#)
- [RESEQUENCE_COLLECTION Procedure](#)
- [RESET_COLLECTION_CHANGED Procedure](#)
- [RESET_COLLECTION_CHANGED_ALL Procedure](#)
- [SORT_MEMBERS Procedure](#)
- [TRUNCATE_COLLECTION Procedure](#)

- [UPDATE_MEMBER Procedure](#)
- [UPDATE_MEMBERS Procedure](#)
- [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](#)
- [UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6](#)

15.1 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 Oracle APEX was installed in Oracle Database 12c and with `MAX_STRING_SIZE=EXTENDED`, then the tables for the APEX 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.

- [Accessing a Collection](#)
- [Determining Collection Status](#)
- [Clearing Collection Session State](#)

15.1.1 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:

<code>COLLECTION_NAME</code>	<code>NOT NULL VARCHAR2(255)</code>
<code>SEQ_ID</code>	<code>NOT NULL NUMBER</code>

C001	VARCHAR2 (4000)
C002	VARCHAR2 (4000)
C003	VARCHAR2 (4000)
C004	VARCHAR2 (4000)
C005	VARCHAR2 (4000)
...	
C050	VARCHAR2 (4000)
N001	NUMBER
N002	NUMBER
N003	NUMBER
N004	NUMBER
N005	NUMBER
D001	DATE
D002	DATE
D003	DATE
D004	DATE
D005	DATE
CLOB001	CLOB
BLOB001	BLOB
XMLTYPE001	XMLTYPE
MD5_ORIGINAL	VARCHAR2 (4000)

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

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

15.1.2 Determining Collection Status

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

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

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

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

 **See Also:**

- "[RESET_COLLECTION_CHANGED Procedure](#)"
- "[COLLECTION_HAS_CHANGED Function](#)"
- "[GET_MEMBER_MD5 Function](#)"

15.1.3 Clearing Collection Session State

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

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

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

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

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

 **See Also:**

- [TRUNCATE_COLLECTION Procedure](#)

15.2 ADD_MEMBER Procedure

Adds a new member to an existing collection. An error occurs if the specified collection does not exist for the current user in the same session for the current application ID.

Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) adds the new member with a sequence ID of 9.

Syntax

```
APEX_COLLECTION.ADD_MEMBER (  
    p_collection_name    IN VARCHAR2,  
    p_c001                IN VARCHAR2 DEFAULT NULL,  
    ...  
    p_c050                IN VARCHAR2 DEFAULT NULL,  
    p_n001                IN NUMBER    DEFAULT NULL,  
    p_n002                IN NUMBER    DEFAULT NULL,  
    p_n003                IN NUMBER    DEFAULT NULL,
```

```

p_n004          IN NUMBER  DEFAULT NULL,
p_n005          IN NUMBER  DEFAULT NULL,
p_d001          IN DATE    DEFAULT NULL,
p_d002          IN DATE    DEFAULT NULL,
p_d003          IN DATE    DEFAULT NULL,
p_d004          IN DATE    DEFAULT NULL,
p_d005          IN DATE    DEFAULT NULL,
p_clob001       IN CLOB    DEFAULT empty_clob(),
p_blob001       IN BLOB    DEFAULT empty_blob(),
p_xmltype001   IN XMLTYPE DEFAULT NULL,
p_generate_md5 IN VARCHAR2 DEFAULT 'NO' )

```

Parameters

 **Note:**

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

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

Example

```

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](#)

15.3 ADD_MEMBER Function

Adds a new member to an existing collection. Calling this function returns the sequence ID of the newly added member. An error occurs 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_xmlype001            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.

Parameter	Description
p_collection_name	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
p_c001 through p_c050	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.

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

Example

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

 **See Also:**

- [GET_MEMBER_MD5 Function](#)

15.4 ADD_MEMBERS Procedure

Adds an array of members to a collection. An error occurs 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 occurs.

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.
The number of members added is based on the number of elements in the first array.

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

Example

The following example adds two new members to the EMPLOYEE table.

```

BEGIN
    APEX_COLLECTION.ADD_MEMBERS (
        p_collection_name => 'EMPLOYEE',
        p_c001 => l_arr1,
        p_c002 => l_arr2);
END;

```

 **See Also:**

- [GET_MEMBER_MD5 Function](#)

15.5 COLLECTION_EXISTS Function

Determines if a collection exists. Returns TRUE if the specified collection exists for the current user in the current session for the current application ID, otherwise FALSE.

Syntax

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

Parameters

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

Example

The following example determines if the collection named EMPLOYEES exists.

```
BEGIN
    l_exists := APEX_COLLECTION.COLLECTION_EXISTS (
        p_collection_name => 'EMPLOYEES');
END;
```

15.6 COLLECTION_HAS_CHANGED Function

Determines if a collection has changed since it was created or since the collection changed flag was reset.

Syntax

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

Parameters

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

Example

The following example determines if the EMPLOYEES collection has changed since it was created or last reset.

```
BEGIN
    l_exists := APEX_COLLECTION.COLLECTION_HAS_CHANGED (
        p_collection_name => 'EMPLOYEES');
END;
```

15.7 COLLECTION_MEMBER_COUNT Function

Retrieves 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, no error occurs.

Syntax

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

Parameters

Parameter	Description
p_collection_name	The name of the collection.

Example

This example retrieves the total number of members in the DEPARTMENTS collection.

```
BEGIN
    l_count := APEX_COLLECTION.COLLECTION_MEMBER_COUNT( p_collection_name =>
'DEPARTMENTS' );
END;
```

15.8 CREATE_COLLECTION Procedure

Creates 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 occurs.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION (
    p_collection_name      IN VARCHAR2,
    p_truncate_if_exists  IN VARCHAR2 DEFAULT 'NO' )
```

Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_truncate_if_exists	If YES, then members of the collection are first truncated if the collection exists and no error occurs. If NO (or not YES), and the collection exists, an error occurs.

Example

This example creates an empty collection named EMPLOYEES.

```
BEGIN
    APEX_COLLECTION.CREATE_COLLECTION(
        p_collection_name => 'EMPLOYEES');
END;
```

See Also:

- [GET_MEMBER_MD5 Function](#)

15.9 CREATE_COLLECTION_FROM_QUERY Procedure

Creates 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 populate 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 occurs.

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

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

Parameter	Description
p_query	Query to execute to populate the members of the collection.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.
p_truncate_if_exists	If YES, then members of the collection are first truncated if the collection exists and no error occurs. If NO (or not YES), and the collection exists, an error occurs.

Example

The following example creates a collection named AUTO and populates it with data from the AUTOS table. Because p_generate_md5 is YES, the MD5 checksum is computed to enable 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](#)

15.10 CREATE_COLLECTION_FROM_QUERY2 Procedure

Creates 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

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

Example

The following example creates a collection named EMPLOYEE and populates 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, ename, job, mgr from emp',
        p_generate_md5 => 'NO');
END;
```

See Also:

- [GET_MEMBER_MD5 Function](#)

15.11 CREATE_COLLECTION_FROM_QUERY_B Procedure

Creates 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 occurs.

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

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute to populate the members of the collection.
p_names	Array of bind variable names used in the query statement.
p_values	Array of bind variable values used in the bind variables in the query statement.
p_max_row_count	Maximum number of rows returned from the query in p_query to add to the collection.
p_truncate_if_exists	If YES, then members of the collection are truncated first if the collection exists and no error occurs. If NO (or not YES), and the collection exists, an error occurs.

Example

The following example creates a collection named EMPLOYEES and populates it with data from the EMP table.

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



See Also:

- [GET_MEMBER_MD5 Function](#)

15.12 CREATE_COLLECTION_FROM_QUERY_B Procedure (No bind version)

Creates 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 occurs.

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 occurs 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

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

Example

The following example creates a collection named EMPLOYEES and populates 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](#)

15.13 CREATE_COLLECTION_FROM_QUERYB2 Procedure

Creates 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 occurs.

This procedure is identical to `CREATE_COLLECTION_FROM_QUERY_B` except the first five columns of the `SELECT` clause must be numeric and the next five columns must be a 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

Parameter	Description
<code>p_collection_name</code>	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.
<code>p_query</code>	Query to execute to populate the members of the collection.
<code>p_names</code>	Array of bind variable names used in the query statement.
<code>p_values</code>	Array of bind variable values used in the bind variables in the query statement.
<code>p_max_row_count</code>	Maximum number of rows returned from the query in <code>p_query</code> to be added to the collection.
<code>p_truncate_if_exists</code>	If YES, then members of the collection are first truncated if the collection exists. If NO (or not YES), and the collection exists, an error occurs.

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;
```

See Also:

- [GET_MEMBER_MD5 Function](#)

15.14 CREATE_COLLECTION_FROM_QUERYB2 Procedure (No bind version)

Creates a collection from a supplied query using bulk operations.

This method offers significantly faster performance than the CREATE_COLLECTION_FROM_QUERY_B 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 occurs.

This procedure is identical to CREATE_COLLECTION_FROM_QUERY_B except the first five columns of the SELECT clause must be numeric and the next five columns must be a 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 0
)

```

Parameters

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

Example

The following example creates a collection named EMPLOYEES and populates 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 dates. 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, ename, job, mgr from emp where deptno = 10';
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
        p_collection_name => 'EMPLOYEES',
        p_query => l_query);
END;

```

See Also:

- [GET_MEMBER_MD5 Function](#)

15.15 CREATE_OR_TRUNCATE_COLLECTION Procedure

Creates 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 (the named collection is truncated).

Syntax

```

APEX_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION (
    p_collection_name   IN VARCHAR2 )

```

Parameters

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

Example

This example removes all members in an existing collection named EMPLOYEES.

```
BEGIN
    APEX_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
        p_collection_name => 'EMPLOYEES');
END;
```

 **See Also:**

- [GET_MEMBER_MD5 Function](#)

15.16 DELETE_ALL_COLLECTIONS Procedure

Use this procedure to delete all collections that belong to the current user in the current Oracle APEX 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;
```

15.17 DELETE_ALL_COLLECTIONS_SESSION Procedure

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

Syntax

```
APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
```

Parameters

None.

Example

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

```
BEGIN
    APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
END;
```

15.18 DELETE_COLLECTION Procedure

Deletes 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 occurs.

Syntax

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

Parameters

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

Example

This example removes the EMPLOYEE collection.

```
BEGIN
    APEX_COLLECTION.DELETE_COLLECTION(
        p_collection_name => 'EMPLOYEE');
END;
```

15.19 DELETE_MEMBER Procedure

Deletes 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 occurs.

Syntax

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

Parameters

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

Example

This example removes the member with a sequence ID of 2 from the EMPLOYEES collection.

```
BEGIN
    APEX_COLLECTION.DELETE_MEMBER(
        p_collection_name => 'EMPLOYEES',
        p_seq => '2');
END;
```

15.20 DELETE_MEMBERS Procedure

Deletes 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 occurs.

If the attribute number specified is invalid or outside the range of 1 to 50, an error occurs.

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

Syntax

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

Parameters

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

Example

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

```
BEGIN
    apex_collection.delete_members(
        p_collection_name => 'GROCERIES'
        p_attr_number      => 5,
        p_attr_value       => 'APPLE' );
    COMMIT;
END;
```

15.21 GET_MEMBER_MD5 Function

Computes and returns 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. The result of this function can 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 occurs.

If the member specified by sequence ID p_seq does not exist, an application error occurs.

Syntax

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

Parameters

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

Parameter	Description
p_seq	Sequence ID of the collection member.

Example

The following example computes the MD5 for the 10th member of the GROCERIES collection.

```
DECLARE
    l_md5 varchar2(4000);
BEGIN
    l_md5 := apex_collection.get_member_md5(
        p_collection_name => 'GROCERIES'
        p_seq              => 10 );
END;
```

 **See Also:**

- [COLLECTION_HAS_CHANGED Function](#)
- [RESET_COLLECTION_CHANGED Procedure](#)
- [RESET_COLLECTION_CHANGED_ALL Procedure](#)

15.22 MERGE_MEMBERS Procedure

Merges 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 two members are merged.

If p_c001 is NULL, an application error occurs.

Syntax

```
APEX_COLLECTION.MERGE_MEMBERS (
    p_collection_name   IN VARCHAR2,
    p_seq                IN apex_application_global.vc_arr2 DEFAULT
empty_vc_arr,
    p_c001               IN apex_application_global.vc_arr2 DEFAULT
empty_vc_arr,
```

```

    p_c002           IN apex_application_global_vc_arr2 DEFAULT
empty_vc_arr,
    p_c003           IN apex_application_global_vc_arr2 DEFAULT
empty_vc_arr,
    ...
    p_c050           IN apex_application_global_vc_arr2 DEFAULT
empty_vc_arr,
    p_null_index     IN NUMBER      DEFAULT 1,
    p_null_value     IN VARCHAR2   DEFAULT NULL,
    p_init_query     IN VARCHAR2   DEFAULT NULL )

```

Parameters

 **Note:**

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

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. Collection names are not case-sensitive and are converted to upper case.
p_c001 through p_c050	Array of attribute values to be merged. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. The count of the p_c001 array is used across all arrays. If no values are provided, then no actions are performed.
p_c0xx	Attribute of NN attributes values to be merged. Maximum length can be 4,000 bytes. The attribute value is truncated to 4,000 bytes if greater than this amount.
p_seq	Identifies the sequence number of the collection member to be merged.
p_null_index	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. The merge function then ignores this row. This results in removing NULL rows 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.
p_null_value	Used with the p_null_index argument. Identifies the NULL value. If used, this value must not be NULL. A typical value for this argument is 0.
p_init_query	If the collection does not exist, the collection is created using this query.

Example

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

```

DECLARE
    l_seq    APEX_APPLICATION_GLOBAL_VC_ARR2;
    l_c001   APEX_APPLICATION_GLOBAL_VC_ARR2;
    l_c002   APEX_APPLICATION_GLOBAL_VC_ARR2;
    l_c003   APEX_APPLICATION_GLOBAL_VC_ARR2;

```

```

BEGIN
    l_seq(1) := 1;
    l_c001(1) := 7369;
    l_c002(1) := 'SMITH';
    l_c003(1) := 'MANAGER';
    l_seq(2) := 2;
    l_c001(2) := 7499;
    l_c002(2) := 'ALLEN';
    l_c003(2) := 'CLERK';

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

```

15.23 MOVE_MEMBER_DOWN Procedure

Adjusts the sequence ID of a 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 occurs.

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

Parameter	Description
<code>p_collection_name</code>	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.
<code>p_seq</code>	Identifies the sequence number of the collection member to be moved down by one.

Example

This example shows how to move 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;
```

15.24 MOVE_MEMBER_UP Procedure

Adjusts 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 occurs.

If the member specified by sequence ID *p_seq* does not exist, an application error occurs.

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

Parameter	Description
<i>p_collection_name</i>	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.
<i>p_seq</i>	Identifies the sequence number of the collection member to be moved up by one.

Example

This example moves a member of the EMPLOYEES collection up 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;
```

15.25 RESEQUENCE_COLLECTION Procedure

For a named collection, updates 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 occurs.

Syntax

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

Parameters

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

Example

This example resequences the `DEPARTMENTS` collection to remove gaps in the sequence IDs.

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

See Also:

- [MOVE_MEMBER_DOWN Procedure](#)
- [MOVE_MEMBER_UP Procedure](#)

15.26 RESET_COLLECTION_CHANGED Procedure

Resets 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 occurs.

Syntax

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

Parameters

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

Example

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

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

15.27 RESET_COLLECTION_CHANGED_ALL Procedure

Resets 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 resets the changed flag for all collections in the user's current session.

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

15.28 SORT_MEMBERS Procedure

Reorders 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 or 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 occurs.

Syntax

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

Parameters

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

Example

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

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

15.29 TRUNCATE_COLLECTION Procedure

Removes 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 occurs.

Syntax

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

Parameters

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

Example

This example removes all members from the DEPARTMENTS collection.

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

 **See Also:**

[CREATE_OR_TRUNCATE_COLLECTION Procedure](#)

15.30 UPDATE_MEMBER Procedure

Updates 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 occurs.

If the member specified by sequence ID `p_seq` does not exist, an application error occurs.

 **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.

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

Example

This example updates the second member of the collection Departments so that the first member attribute becomes Engineering and the second member attribute becomes Sales.

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

15.31 UPDATE_MEMBERS Procedure

Updates 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 occurs.

The count of elements in the p_seq PL/SQL table is used as the total number of items across all PL/SQL tables. If p_seq.count = 2 and p_c001.count = 10, only two members are updated.

If p_seq is NULL, an application error occurs.

If the member specified by sequence ID p_seq does not exist, an application error occurs.

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.

Parameter	Description
p_collection_name	The name of the collection to update. Maximum length is 255 bytes. Collection names are not case-sensitive and are converted to uppercase.
p_seq	Array of member sequence IDs to be updated. The count of the p_seq array is used across all arrays.
p_c001 through p_c050	Array of attribute values to be updated.
p_n001 through p_n005	Array of numeric attribute values to be updated.
p_d001 through p_d005	Array of date attribute values to be updated.

Example

```
DECLARE
    l_seq   apex_application_global_vc_arr2;
    l_carr  apex_application_global_vc_arr2;
    l_narr  apex_application_global_n_arr;
    l_darr  apex_application_global_d_arr;
BEGIN
    l_seq(1)  := 10;
    l_seq(2)  := 15;
    l_carr(1) := 'Apples';
```

```

l_carr(2) := 'Grapes';
l_narr(1) := 100;
l_narr(2) := 150;
l_darr(1) := sysdate;
l_darr(2) := sysdate;

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

```

15.32 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1

Updates 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 occurs.

If the member specified by sequence ID `p_seq` does not exist, an application error occurs.

If the attribute number specified is invalid or outside the range 1-50, an error occurs.

Any attribute value exceeding 4,000 bytes are truncated to 4,000 bytes.

Syntax

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

Parameters

Note:

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

Parameter	Description
<code>p_collection_name</code>	The name of the collection. Maximum length is 255 bytes. Collection names are case-insensitive and are 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.
<code>p_seq</code>	Sequence ID of the collection member to be updated.
<code>p_attr_number</code>	Attribute number of the member attribute to be updated. Valid values are 1 through 50. Any number outside of this range is ignored.
<code>p_attr_value</code>	Attribute value of the member attribute to be updated.

Example

This example updates the second member of the collection `Departments`, updating the first member attribute to `Engineering`.

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

15.33 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2

Updates 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 occurs.

If the member specified by sequence ID `p_seq` does not exist, an application error occurs.

If the attribute number specified is invalid or outside the valid range (currently only 1 for CLOB), an error occurs.

Syntax

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

Parameters

 **Note:**

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

Parameter	Description
<code>p_collection_name</code>	The name of the collection. Maximum length can be 255 bytes. Collection names are case-insensitive and are 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.
<code>p_seq</code>	Sequence ID of the collection member to be updated.
<code>p_clob_number</code>	Attribute number of the CLOB member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.
<code>p_clob_value</code>	Attribute value of the CLOB member attribute to be updated.

Example

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

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

15.34 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3

Updates 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 occurs.

If the member specified by sequence ID `p_seq` does not exist, an application error occurs.

If the attribute number specified is invalid or outside the valid range (currently only 1 for BLOB), an error occurs.

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.

Parameter	Description
<code>p_collection_name</code>	The name of the collection. Maximum length is 255 bytes. Collection names are case-insensitive and are 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.
<code>p_seq</code>	Sequence ID of the collection member to be updated.
<code>p_blob_number</code>	Attribute number of the BLOB member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.
<code>p_blob_value</code>	Attribute value of the BLOB member attribute to be updated.

Example

The following example sets the first and only BLOB attribute of collection sequence number 2 in the collection `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;
```

15.35 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4

Updates 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 occurs.

If the member specified by sequence ID `p_seq` does not exist, an application error occurs.

If the attribute number specified is invalid or outside the valid range (currently only 1 for XMLTYPE), an error occurs.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name    IN VARCHAR2,
    p_seq                 IN NUMBER,
    p_xmлtype_number     IN NUMBER,
    p_xmлtype_value      IN SYS.XMLTYPE )
```

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.

Parameter	Description
<code>p_collection_name</code>	The name of the collection. Maximum length can be 255 bytes. Collection names are case-insensitive and are 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.
<code>p_seq</code>	Sequence ID of the collection member to be updated.
<code>p_xmлtype_number</code>	Attribute number of the XMLTYPE member attribute to be updated. Valid value is 1. Any number outside of this range is ignored.
<code>p_xmлtype_value</code>	Attribute value of the XMLTYPE member attribute to be updated.

Example

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

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

15.36 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5

Updates 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 occurs.

If the member specified by sequence ID `p_seq` does not exist, an application error occurs.

If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for NUMBER), an error occurs.

Syntax

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

Parameters

 **Note:**

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

Parameter	Description
<code>p_collection_name</code>	The name of the collection. Maximum length can be 255 bytes. Collection names are case-insensitive and are 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.
<code>p_seq</code>	Sequence ID of the collection member to be updated.
<code>p_attr_number</code>	Attribute number of the NUMBER member attribute to be updated. Valid value is 1 through 5. Any number outside of this range is ignored.
<code>p_number_value</code>	Attribute value of the NUMBER member attribute to be updated.

Example

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

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

15.37 UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6

Updates 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 occurs.

If the member specified by sequence ID p_seq does not exist, an application error occurs.

If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for DATE), an error occurs.

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.

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

Example

This example updates the first date attribute of the second collection member in collection named Departments and sets 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;
```

16

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](#)
- [CREATE_CREDENTIAL Procedure Signature 1](#)
- [CREATE_CREDENTIAL Procedure Signature 2](#)
- [DROP_CREDENTIAL Procedure](#)
- [SET_ALLOWED_URLS Procedure](#)
- [SET_DATABASE_CREDENTIAL Procedure](#)
- [SET_PERSISTENT_CREDENTIALS Procedure Signature 1](#)
- [SET_PERSISTENT_CREDENTIALS Procedure Signature 2](#)
- [SET_PERSISTENT_CREDENTIALS Procedure Signature 3](#)
- [SET_PERSISTENT_TOKEN Procedure](#)
- [SET_SCOPE Procedure](#)
- [SET_SESSION_CREDENTIALS Procedure Signature 1](#)
- [SET_SESSION_CREDENTIALS Procedure Signature 2](#)
- [SET_SESSION_CREDENTIALS Procedure Signature 3](#)
- [SET_SESSION_TOKEN Procedure](#)

16.1 CLEAR_TOKENS Procedure

This procedure clears all acquired tokens for the provided credential.

Only useful for OAuth-based flows.

Syntax

```
PROCEDURE CLEAR_TOKENS( p_credential_static_id IN VARCHAR2 );
```

Parameters

Parameters	Description
p_credential_static_id	The credential static ID.

Example

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

```
BEGIN  
    apex_credential.clear_tokens(
```

```

    p_credential_static_id => 'OAuth Login' );
END;

```

16.2 CREATE_CREDENTIAL Procedure Signature 1

This procedure creates a credential definition.

Syntax

```

PROCEDURE CREATE_CREDENTIAL (
    p_credential_name      IN VARCHAR2,
    p_credential_static_id IN VARCHAR2,
    p_authentication_type  IN VARCHAR2,
    p_scope                IN VARCHAR2          DEFAULT NULL,
    p_allowed_urls         IN apex_t_varchar2 DEFAULT NULL,
    p_prompt_on_install    IN BOOLEAN           DEFAULT FALSE,
    p_credential_comment   IN VARCHAR2          DEFAULT NULL )

```

Parameters

Parameter	Description
p_credential_name	The credential name.
p_credential_static_id	The credential static ID.
p_authentication_type	The authentication type. Supported types: <ul style="list-style-type: none"> • APEX_CREDENTIAL.C_TYPE_BASIC • APEX_CREDENTIAL.C_TYPE_OAUTH_CLIENT_CRED • APEX_CREDENTIAL.C_TYPE_JWT • APEX_CREDENTIAL.C_TYPE_OCI • APEX_CREDENTIAL.C_TYPE_HTTP_HEADER • APEX_CREDENTIAL.C_TYPE_HTTP_QUERY_STRING
p_scope	(Optional) OAuth 2.0 scope.
p_allowed_urls	(Optional) List of URLs (as APEX_T_VARCHAR2) that these credentials can access.
p_prompt_on_install	(Optional) Choose whether prompts for this credential should be displayed when the application is being imported on another Oracle APEX instance.
p_credential_comment	(Optional) Credential comment.

Example

The following example creates a credential definition "OAuth Login."

```

BEGIN
    -- first set the workspace
    apex_util.set_workspace(p_workspace => 'MY_WORKSPACE');

    apex_credential.create_credential (
        p_credential_name => 'OAuth Login',
        p_credential_static_id => 'OAUTH_LOGIN',
        p_authentication_type => apex_credential.C_TYPE_OAUTH_CLIENT_CRED,
        p_scope => 'email',
        p_allowed_urls => apex_t_varchar2( 'https://tokenserver.example.com/

```

```

oauth2/token', 'https://www.oracle.com' ),
    p_prompt_on_install => false,
    p_credential_comment => 'Credential for OAuth Login' );

-- should be followed by set_persistent_credentials
apex_credential.set_persistent_credentials (
    p_credential_static_id => 'OAUTH_LOGIN',
    p_client_id => 'dnkjfq237o8832ndj98098-..',
    p_client_secret => '1278672tjkgsaGSDA789312..');
END;

```

16.3 CREATE_CREDENTIAL Procedure Signature 2

This procedure creates a credential definition.

Syntax

```

PROCEDURE CREATE_CREDENTIAL (
    p_credential_name          IN VARCHAR2,
    p_credential_static_id     IN VARCHAR2,
    p_authentication_type       IN VARCHAR2,
    p_scope                     IN VARCHAR2        DEFAULT NULL,
    p_allowed_urls              IN apex_t_varchar2 DEFAULT NULL,
    p_prompt_on_install         IN BOOLEAN        DEFAULT FALSE,
    p_credential_comment        IN VARCHAR2        DEFAULT NULL,
    --
    p_db_credential_name        IN VARCHAR2        DEFAULT NULL,
    p_db_credential_is_instance IN BOOLEAN        DEFAULT NULL )

```

Parameters

Parameter	Description
p_credential_name	The credential name.
p_credential_static_id	The credential static ID.
p_authentication_type	The authentication type. Supported types: <ul style="list-style-type: none"> • APEX_CREDENTIAL.C_TYPE_BASIC • APEX_CREDENTIAL.C_TYPE_OAUTH_CLIENT_CRED • APEX_CREDENTIAL.C_TYPE_JWT • APEX_CREDENTIAL.C_TYPE_OCI • APEX_CREDENTIAL.C_TYPE_HTTP_HEADER • APEX_CREDENTIAL.C_TYPE_HTTP_QUERY_STRING
p_scope	OAuth 2.0 scope.
p_allowed_urls	List of URLs (as APEX_T_VARCHAR2) that these credentials can access.
p_db_credential_name	Name of the database credential to be referenced.
p_db_credential_is_instance	Whether the database credential was made available at the Oracle APEX instance level (all workspaces). This parameter can only be used when instance credentials are enabled for the APEX instance using the INSTANCE_DBMS_CREDENTIAL_ENABLED instance parameter.
p_prompt_on_install	Choose whether prompts for this credential should be displayed when the application is being imported on another APEX instance.

Parameter	Description
p_credential_comment	Credential comment.

Example

The following example creates a new web credential "OAuth Login."

```
BEGIN
    -- set the workspace
    apex_util.set_workspace(p_workspace => 'MY_WORKSPACE');

    apex_credential.create_credential (
        p_credential_name      => 'OAuth Login',
        p_credential_static_id => 'OAUTH_LOGIN',
        p_authentication_type  => apex_credential.c_type_oauth_client_cred,
        p_scope                => 'email',
        p_allowed_urls         => apex_t_varchar2( 'https://
tokenserwer.mycompany.com/oauth2/token', 'https://www.oracle.com' ),
        p_prompt_on_install    => false,
        p_credential_comment   => 'Credential for OAuth Login' );

    -- store client ID and client secret into the credential
    apex_credential.set_persistent_credentials (
        p_credential_static_id => 'OAUTH_LOGIN',
        p_client_id            => 'dnkjg237o8832ndj98098-..',
        p_client_secret         => '1278672tjkksaGSDA789312..');
END;
```

16.4 DROP_CREDENTIAL Procedure

This procedure drops a credential definition.

Syntax

```
APEX_CREDENTIAL.DROP_CREDENTIAL (
    p_credential_static_id  IN VARCHAR2 )
```

Parameters

Parameter	Description
p_credential_static_id	The credential static ID.

Example

The following example drops the credential definition "OAuth Login."

```
BEGIN
    -- first set the workspace
    apex_util.set_workspace(p_workspace => 'MY_WORKSPACE');

    apex_credential.drop_credential (
```

```

    p_credential_static_id => 'OAUTH_LOGIN' );
END;

```

16.5 SET_ALLOWED_URLS Procedure

This procedure sets a list of URLs that can be used for this credential.

A credential can be used for a HTTP request if its target URL matches one of the URLs in this list. Matching is done on a starts-with basis.

For instance, if "https://www.oracle.com" and "https://apex.oracle.com/ords/" are set as the allowed URLs, then the credential can be used for the following HTTP request examples:

- `https://www.oracle.com/`
- `https://www.oracle.com/myrest/service`
- `https://apex.oracle.com/ords/secret/workspace`

The credential cannot be used for the following request examples:

- `https://web.oracle.com`
- `https://apex.oracle.com/apex/workspace`
- `http://www.oracle.com/`

For security, the credential secret (Client Secret, Password, Private Key) must be passed in too. If not passed, or passed as NULL, the secret is cleared.

Syntax

```

PROCEDURE SET_ALLOWED_URLS (
    p_credential_static_id  IN VARCHAR2,
    p_allowed_urls          IN apex_t_varchar2,
    p_client_secret          IN VARCHAR2 );

```

Parameters

Parameter	Description
<code>p_credential_static_id</code>	The credential static ID.
<code>p_allowed_urls</code>	List of URLs (as <code>APEX_T_VARCHAR2</code>) that these credentials can access.
<code>p_client_secret</code>	Client Secret. If allowed URLs are changed, this must be provided again.

Examples

This example sets allowed URLs for the credential OAuth Login.

```

BEGIN
    apex_credential.set_allowed_urls (
        p_credential_static_id => 'OAuth Login',
        p_allowed_urls          => apex_t_varchar2(
            'https://tokenserver.example.com/oauth2/token',
            'https://www.oracle.com' ),

```

```

    p_client_secret      => '1278672tjksaGSDA789312..' );
END;

```

16.6 SET_DATABASE_CREDENTIAL Procedure

This procedure updates database credential properties for a web credential.

If a web credential references a database credential, then it does not store secrets itself - that is done by the database credential. See DBMS_CREDENTIAL for more information.

Clears all existing client IDs, client secrets, all tokens, and the "Valid For URL" attribute. If database credentials for HTTP requests are not supported on the database, and the credential did not reference a database credential before, this procedure raises an error.

Syntax

```
APEX_CREDENTIAL.SET_DATABASE_CREDENTIAL (
    p_credential_static_id      IN VARCHAR2,
    p_db_credential_name        IN VARCHAR2,
    p_db_credential_is_instance IN BOOLEAN DEFAULT FALSE )
```

Parameters

Parameter	Description
p_credential_static_id	The credential static ID.
p_db_credential_name	Name of the database credential to be referenced.
p_db_credential_is_instance	Whether the database credential was made available at the Oracle APEX instance level (all workspaces). This parameter can only be used when instance credentials are enabled for the APEX instance using the INSTANCE_DBMS_CREDENTIAL_ENABLED instance parameter.

Example

The following example changes the referenced database credential to USE_THIS_DB_CREDENTIAL.

```

BEGIN
    -- set the workspace
    apex_util.set_workspace(p_workspace => 'MY_WORKSPACE');

    -- change the referenced database credential
    apex_credential.set_database_credential (
        p_credential_static_id => 'OAUTH_LOGIN',
        p_db_credential_name   => 'USE_THIS_DB_CREDENTIAL' );
END;

```

See Also:

DBMS_CREDENTIAL in *Oracle Database PL/SQL Packages and Types Reference*

16.7 SET_PERSISTENT_CREDENTIALS Procedure Signature 1

This procedure sets provided credential attributes persistently, beyond the current session. Already stored access, refresh or ID tokens for the provided credential are removed.

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. Stored access, refresh or ID tokens for that credential, will be deleted.

Syntax

```
PROCEDURE SET_PERSISTENT_CREDENTIALS (
    p_credential_static_id  IN VARCHAR2,
    p_client_id              IN VARCHAR2,
    p_client_secret           IN VARCHAR2,
    p_namespace                IN VARCHAR2 DEFAULT NULL,
    p_fingerprint             IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameters	Description
p_credential_static_id	Credential static ID.
p_client_id	Use Client ID for OAuth Credentials. Use User OCID for OCI Credentials.
p_client_secret	Use Client Secret for OAuth Credentials. Use Private Key for OCI Credentials.
p_namespace	Use the Tenancy OCID for OCI Credentials.
p_fingerprint	Use the Public Key Fingerprint for OCI Credentials.

Example 1

The following example sets credential attributes for OAuth Login.

```
BEGIN
    apex_credential.set_persistent_credentials (
        p_credential_static_id  => 'OAuth Login',
        p_client_id              => 'dnkjg237o8832ndj98098-..',
        p_client_secret           => '1278672tjksaGSDA789312..');
END;
```

Example 2

The following example sets credential attributes for OCI Login.

```
BEGIN
    apex_credential.set_persistent_credentials (
        p_credential_static_id  => 'OCI Login',
        p_client_id              => 'ocid1.user.oc1..',
        p_client_secret           => 'MIIEowIBAAKCAQEAsjhTVL..',
        p_namespace                => 'ocid1.tenancy.oc1..');
```

```

    p_fingerprint      => 'ff:ff:ee:00:...' );
END;

```

16.8 SET_PERSISTENT_CREDENTIALS Procedure Signature 2

This procedure sets user name and password for the provided credential persistently, beyond the current session. Typically used for BASIC authentication.

Syntax

```

PROCEDURE SET_PERSISTENT_CREDENTIALS (
    p_credential_static_id  IN VARCHAR2,
    p_username              IN VARCHAR2,
    p_password              IN VARCHAR2 );

```

Parameters

Parameters	Description
p_credential_static_id	Credential static ID.
p_username	Credential user name.
p_password	Credential password.

Example

The following example sets user name and password into credential `Login` persistently.

```

BEGIN
    apex_credential.set_persistent_credentials (
        p_credential_static_id => 'Login',
        p_username             => 'scott',
        p_password              => 'tiger );
END;

```

16.9 SET_PERSISTENT_CREDENTIALS Procedure Signature 3

This procedure sets provided credential attributes persistently beyond the current Oracle APEX session.

Syntax

```

APEX_CREDENTIAL.SET_PERSISTENT_CREDENTIALS (
    p_credential_static_id  IN VARCHAR2,
    p_key                  IN VARCHAR2,
    p_value                IN VARCHAR2 )

```

Parameters

Parameter	Description
p_credential_static_id	Credential static ID.

Parameter	Description
p_key	Credential key (for example, HTTP Header or Cookie name).
p_value	Credential value.

Example

The following example sets attributes into credential my_API_key persistently.

```
BEGIN
    apex_credential.set_persistent_credentials (
        p_credential_static_id => 'my_API_key',
        p_key => 'api_key',
        p_value => 'lsjkgjw4908902ru9fj879q367891hdaw' );
END;
```

16.10 SET_PERSISTENT_TOKEN Procedure

This procedure sets a token into the provided credential persistently, beyond the current Oracle APEX session. The token is encrypted for security. Client ID and Client Secret are **not** stored in the credential store by this procedure.

Syntax

```
APEX_CREDENTIAL.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

Parameters	Description
p_credential_static_id	The credential static ID.
p_token_type	One of the constants C_TOKEN_ACCESS, C_TOKEN_REFRESH, or C_TOKEN_ID.
p_token_value	The token value.
p_token_expires	The token expiry date.

Example

The following example stores the OAuth2 access token with value sdakjjkhw7632178jh12hs876e38.. and expiry date of 2023-10-31 into the credential OAuth Login.

```
BEGIN
    apex_credential.set_persistent_token (
        p_credential_static_id => 'OAuth Login',
        p_token_type => apex_credential.C_TOKEN_ACCESS,
        p_token_value => 'sdakjjkhw7632178jh12hs876e38..',
```

```

    p_token_expires => to_date('2023-10-31', 'YYYY-MM-DD') );
END;

```

16.11 SET_SCOPE Procedure

This procedure changes the "scope" attribute of a Web Credential. All existing tokens for the given credential are cleared.

Syntax

```
APEX_CREDENTIAL.SET_SCOPE (
    p_credential_static_id      IN VARCHAR2,
    p_scope                     IN VARCHAR2 );
```

Parameters

Parameter	Description
p_credential_static_id	Credential static ID.
p_scope	New scope value to store within the Web Credential.

Example

The following example sets allowed URLs for the credential "OAuth_Login."

```
BEGIN
    apex_credential.set_scope (
        p_credential_static_id => 'OAuth_Login',
        p_scope                 => 'new-scope-value' );
END;
```

16.12 SET_SESSION_CREDENTIALS Procedure Signature 1

This procedure sets user name and password for the provided credential for the current Oracle APEX session. Typically used for `BASIC` authentication when the end user provides the credentials.

Syntax

```
APEX_CREDENTIAL.SET_SESSION_CREDENTIALS (
    p_credential_static_id  IN VARCHAR2,
    p_username              IN VARCHAR2,
    p_password              IN VARCHAR2 );
```

Parameters

Parameters	Description
p_credential_static_id	The credential static ID.
p_username	The credential user name.
p_password	The credential password.

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;
```

16.13 SET_SESSION_CREDENTIALS Procedure Signature 2

This procedure sets provided credential attributes for the current Oracle APEX session. Typically used for the OAuth2 client credentials or OCI (Oracle Cloud Infrastructure) credential types.

Syntax

```
APEX_CREDENTIAL.SET_SESSION_CREDENTIALS (
    p_credential_static_id  IN VARCHAR2,
    p_client_id              IN VARCHAR2,
    p_client_secret          IN VARCHAR2,
    p_namespace               IN VARCHAR2 DEFAULT NULL,
    p_fingerprint            IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameters	Description
p_credential_static_id	Credential static ID.
p_client_id	Use Client ID for OAuth credentials (use User OCID for OCI credentials).
p_client_secret	Use Client Secret for OAuth credentials (use Private Key for OCI credentials).
p_namespace	Use the Tenancy OCID for OCI credentials.
p_fingerprint	Use the Public Key Fingerprint for OCI credentials.

Example 1

The following example sets credential attributes for OAuth Login.

```
BEGIN
    apex_credential.set_session_credentials (
        p_credential_static_id => 'OAuth Login',
        p_client_id             => 'dnkjg237o8832ndj98098..',
        p_client_secret          => '1278672tjksaGSDA789312..');
END;
```

Example 2

The following example sets the credential attributes for OCI Login.

```
BEGIN
    apex_credential.set_session_credentials (
        p_credential_static_id => 'OCI Login',
        p_client_id            => 'ocid1.user.oc1...',
        p_client_secret         => 'MIIEowIBAAKCAQEAsjhTVL...',
        p_namespace             => 'ocid1.tenancy.oc1...',
        p_fingerprint          => 'ff:ff:ee:00:' );
END;
```

16.14 SET_SESSION_CREDENTIALS Procedure Signature 3

This procedure sets provided credential attributes for the current Oracle APEX session.

Syntax

```
APEX_CREDENTIAL.SET_SESSION_CREDENTIALS (
    p_credential_static_id IN VARCHAR2,
    p_key                  IN VARCHAR2,
    p_value                IN VARCHAR2 );
```

Parameters

Parameter	Description
p_credential_static_id	The credential static ID.
p_key	Credential key (name of the HTTP Header or Query String Parameter).
p_value	Credential secret value.

Example

The following example set attributes into credential `my_API_key` for the current APEX session persistently.

```
BEGIN
    apex_credential.set_session_credentials (
        p_credential_static_id => 'my_API_key',
        p_key                  => 'api_key',
        p_value                => 'lsjkgjw4908902ru9fj879q367891hdaw' );
END;
```

16.15 SET_SESSION_TOKEN Procedure

This procedure sets a token into the provided credential for the duration of the current Oracle APEX session. The token is encrypted and can only be used by the current APEX session. Client ID and Client Secret are **not** stored in the credential by this procedure.

Syntax

```
APEX_CREDENTIAL.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

Parameters	Description
p_credential_static_id	The credential static ID.
p_token_type	One of the constants C_TOKEN_ACCESS, C_TOKEN_REFRESH, or C_TOKEN_ID.
p_token_value	The token value.
p_token_expiry	The token expiry date.

Example

The following example stores the OAuth access token with value sdakjjkhw7632178jh12hs876e38.. and expiry date of 2023-10-31 into the credential OAuth Login.

```
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_expires        => to_date('2023-10-31', 'YYYY-MM-DD') );
END;
```

17

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 \(Deprecated\)](#)
- [ADD_FILE Procedure](#)

17.1 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

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

Example

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

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

17.2 ADD_3RD_PARTY_LIBRARY_FILE Procedure (Deprecated)

This procedure adds the link tag to load a third-party CSS file and also takes into account the specified CDN (content delivery network) for the application.

Supported libraries include:

- jQuery
- jQueryUI

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

Syntax

```
APEX_CSS.ADD_3RD_PARTY_LIBRARY_FILE (
    p_library      IN  VARCHAR2,
    p_file_name    IN  VARCHAR2 DEFAULT NULL,
    p_directory    IN  VARCHAR2 DEFAULT NULL,
    p_version      IN  VARCHAR2 DEFAULT NULL,
    p_media_query  IN  VARCHAR2 DEFAULT NULL,
    p_attributes   IN  VARCHAR2 DEFAULT NULL )
```

Parameters

Parameters	Description
p_library	Use one of the c_library_* constants.
p_file_name	Specifies the file name excluding version, .min, and .css.
p_directory	(Optional) Directory where the file p_file_name is located.
p_version	(Optional) If no value is provided, then uses the same version shipped with APEX.
p_media_query	(Optional) Value that is set as media query.
p_attributes	Extra attributes to add to the link tag.

Note:

Callers are responsible for escaping this parameter.

Example

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

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

17.3 ADD_FILE Procedure

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

Syntax

```
APEX_CSS.ADD_FILE (
    p_name      IN  VARCHAR2,
    p_directory IN  VARCHAR2 DEFAULT
```

```
apex_application.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 is desupported and has no effect
    p_ie_condition  IN      VARCHAR2 DEFAULT NULL,
    p_attributes     IN      VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_name	Name of the CSS file.
p_directory	Begin of the URL where the CSS file should be read from. If you use this function for a plug-in, set this parameter to <code>p_plugin.file_prefix</code>
p_version	Identifier of the version of the CSS file. The version will be added to the CSS filename. In most cases you should use the default of <code>NULL</code> as the value.
p_skip_extension	The function automatically adds <code>.css</code> to the CSS filename. If set to <code>TRUE</code> , the function ignores this addition.
p_media_query	Value set as media query.
p_ie_condition	(Desupported) Condition used as Internet Explorer condition.
p_attributes	Extra attributes to add to the link tag.

Note:

Callers are responsible for escaping this parameter.

Example

Adds the CSS file `jquery.autocomplete.css` in the directory specified by `p_plugin.file_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.file_prefix );
```

18

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](#)
- [LDAP_DNPREP 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](#)

18.1 APPLICATION_PAGE_ITEM_EXISTS Function

This function checks for the existence of a 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

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

Example

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

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

18.2 CURRENT_PAGE_IS_PUBLIC Function

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

Syntax

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

Example

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

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

 See Also:

Editing Page Attributes in *Oracle APEX App Builder User's Guide*.

18.3 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

Parameter	Description
<code>p_user</code>	Login name of the user.
<code>p_session_id</code>	The session ID.

Example

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

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

 See Also:

- [SET_USER Procedure](#)
- [SET_SESSION_ID Procedure](#)

18.4 GET_COOKIE_PROPS Procedure

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

Syntax

```
APEX_CUSTOM_AUTH.GET_COOKIE_PROPS (
    p_app_id           IN      NUMBER,
    p_cookie_name     OUT     VARCHAR2,
```

```

    p_cookie_path          OUT VARCHAR2,
    p_cookie_domain        OUT VARCHAR2
    p_secure                OUT BOOLEAN )
```

Parameters

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

Example

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

```

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

18.5 GET_LDAP_PROPS Procedure

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

Syntax

```

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

Parameters

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

Example

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

```

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

```

18.6 GET_NEXT_SESSION_ID Function

This function generates the next session ID from the Oracle APEX 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;
```

18.7 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;
```

18.8 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;
```

18.9 GET_SESSION_ID_FROM_COOKIE Function

This function returns the Oracle APEX 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;
```

18.10 GET_USER Function

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

Syntax

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

Example

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

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

18.11 GET_USERNAME Function

This function returns user name registered with the current Oracle APEX 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;

```

18.12 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;

```

18.13 LDAP_DNPREP Function

This function replaces any occurrences of a period character (.) with an underscore character (_) in the passed in `p_username` value and then returns that newly massaged username value.

Syntax

```

APEX_CUSTOM_AUTH.LDAP_DNPREP (
    p_username IN VARCHAR2 )
    RETURN VARCHAR2;

IS
BEGIN
    RETURN replace(p_username, '.', '_');
END ldap_dnprep;

```

Parameters

Parameter	Description
<code>p_username</code>	Username value of an end user.

Example

The following example demonstrates how to return a username formatted for LDAP authentication.

```
return apex_custom_auth.ldap_dnprep(p_username =>
:USERNAME);
```

18.14 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 )
```

 **Note:**

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

Parameter

Parameter	Description
<code>p_uname</code>	Login name of the user.
<code>p_password</code>	Clear text user password.
<code>p_session_id</code>	Current Oracle APEX session ID. Do not use bind variable notations for <code>p_session_id</code> argument.
<code>p_app_page</code>	Current application ID. After login page separated by a colon (:).
<code>p_entry_point</code>	Internal use only.
<code>p_preserve_case</code>	If TRUE, do not include <code>p_uname</code> in uppercase during session registration.

Example

The following example performs the user authentication and session registration.

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

```

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

```

18.15 LOGOUT Procedure (Deprecated)

 **Note:**

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

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

Syntax

```

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

```

Parameter

Parameter	Description
<code>p_this_app</code>	Current application ID.
<code>p_next_app_page_sess</code>	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).
<code>p_next_url</code>	URL to redirect to (use this instead of <code>p_next_app_page_sess</code>).

Example

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

```

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

```

18.16 POST_LOGIN Procedure

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

Syntax

```

APEX_CUSTOM_AUTH.POST_LOGIN (
    p_uname       IN VARCHAR2 DEFAULT NULL,
    p_session_id  IN VARCHAR2 DEFAULT NULL,

```

```
p_app_page      IN VARCHAR2 DEFAULT NULL,
p_preserve_case IN BOOLEAN  DEFAULT FALSE )
```

Parameter

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

Example

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

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

18.17 SESSION_ID_EXISTS Function

This function returns a Boolean result based on the global package variable containing the current Oracle APEX session ID.

Returns TRUE if the result is a positive number; returns FALSE if the result is a negative number.

Syntax

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

Example

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

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

18.18 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

Parameter	Description
<code>p_session_id</code>	The session ID to be registered.

Example

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

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

18.19 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;
```

18.20 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

Parameter	Description
p_user	The user ID to be registered.

Example

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

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

19

APEX_DATA_LOADING

The APEX_DATA_LOADING package provides the ability to load data by calling an application data loading definition. This can be used in place of native data loading.

- [Data Types](#)
- [GET_FILE_PROFILE Function](#)
- [LOAD_DATA Function Signature 1](#)
- [LOAD_DATA Function Signature 2](#)

19.1 Data Types

The APEX_DATA_LOADING package uses the following data types.

```
type t_data_load_result is record(
    processed_rows      PLS_INTEGER,
    error_rows          PLS_INTEGER );
```

19.2 GET_FILE_PROFILE Function

This function returns the file profile (determined by the data loading definition) in JSON format.

Syntax

```
APEX_DATA_LOADING.GET_FILE_PROFILE (
    p_application_id    IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_static_id         IN VARCHAR2 )
RETURN CLOB;
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the data load definition.
p_static_id	Static ID of the data loading definition to execute.

Example

This example parses and fetches the first 10 columns using a file uploaded from P1_FILE File Browse item and the file profile computed from the data load defintion.

```
select p.line_number,
       p.col001,
       p.col002,
       p.col003,
       p.col004,
```

```

        p.col005,
        p.col006,
        p.col007,
        p.col008,
        p.col009,
        p.col010
    from apex_application_temp_files          f,
         table( apex_data_parser.parse(
            p_content      => f.blob_content,
            p_file_name    => f.filename,
            p_file_profile => apex_data_loading.get_file_profile
                               ( p_static_id => 'my-load-definition'),
            p_max_rows     => 100 ) ) p
    where f.name = :P1_FILE;

```

19.3 LOAD_DATA Function Signature 1

This function loads file data and returns loading status information containing processed rows and error rows.

Syntax

```
APEX_DATA_LOADING.LOAD_DATA (
    p_application_id    IN NUMBER      DEFAULT apex_application.g_flow_id,
    p_static_id         IN VARCHAR2,
    p_data_to_load      IN BLOB,
    p_xlsx_sheet_name   IN VARCHAR2    DEFAULT NULL )
RETURN t_data_load_result;
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the data load definition.
p_static_id	Static ID of the data loading definition to execute.
p_data_to_load	BLOB file to be loaded.
p_xlsx_sheet_name	For XLSX files, the worksheet to extract.

Example

This example fetches a file (uploaded with the PX_FILEBROWSE_ITEM) from the APEX_APPLICATION_TEMP_FILES table and executes the my-load-definition data loading definition.

```

DECLARE
    l_file blob;
    l_load_result apex_data_loading.t_data_load_result;
BEGIN
    apex_session.create_session( 100, 1, 'ADMIN' );
    SELECT blob_content
        INTO l_file
        FROM apex_application_temp_files
    WHERE name = :PX_FILEBROWSE_ITEM;

```

```

l_load_result := apex_data_loading.load_data (
    p_static_id      => 'my-load-definition',
    p_data_to_load   => l_file );
dbms_output.put_line( 'Processed ' || l_load_result.processed_rows || '
rows.' );
END;

```

19.4 LOAD_DATA Function Signature 2

This function loads CLOB data and returns loading status information containing processed rows and error rows.

Syntax

```

APEX_DATA_LOADING.LOAD_DATA (
    p_application_id   IN NUMBER          DEFAULT apex_application.g_flow_id,
    p_static_id        IN VARCHAR2,
    p_data_to_load     IN CLOB,
    p_xlsx_sheet_name IN VARCHAR2        DEFAULT NULL )
RETURN t_data_load_result;

```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the data load definition.
p_static_id	Static ID of the data loading definition to execute.
p_data_to_load	CLOB data to be loaded.
p_xlsx_sheet_name	For XLSX files, the worksheet to extract.

Example

This example gets data (copy and pasted into the `:PX_DATA` textarea) and executes the `my-load-definition` data loading definition.

```

DECLARE
    l_load_result apex_data_loading.t_data_load_result;
BEGIN
    apex_session.create_session( 100, 1, 'ADMIN' );

    l_load_result := apex_data_loading.load_data (
        p_static_id      => 'my-load-definition',
        p_data_to_load   => :PX_DATA );
    dbms_output.put_line( 'Processed ' || l_load_result.processed_rows || '
rows.' );
END;

```

20

APEX_DATA_EXPORT

The APEX_DATA_EXPORT package contains the implementation to export data from Oracle APEX. Supported filetypes include: PDF, XLSX, HTML, CSV, XML and JSON.

Use the `EXPORT` function to pass a query context from the `APEX_EXEC` package and return the `t_export` type, which includes the contents in a LOB.

- [Global Constants](#)
- [Data Types](#)
- [ADD_AGGREGATE Procedure](#)
- [ADD_COLUMN Procedure](#)
- [ADD_COLUMN_GROUP Procedure](#)
- [ADD_HIGHLIGHT Procedure](#)
- [DOWNLOAD Procedure](#)
- [EXPORT Function](#)
- [GET_PRINT_CONFIG Procedure](#)

20.1 Global Constants

The APEX_DATA_EXPORT package uses the following constants.

Export Format Constants

Constants used in the `EXPORT` function. The `c_format_pxml` and `c_format_pjson` formats are optimized for printing.

<code>c_format_csv</code>	<code>constant t_format</code>	<code>:= 'CSV';</code>
<code>c_format_html</code>	<code>constant t_format</code>	<code>:= 'HTML';</code>
<code>c_format_pdf</code>	<code>constant t_format</code>	<code>:= 'PDF';</code>
<code>c_format_xlsx</code>	<code>constant t_format</code>	<code>:= 'XLSX';</code>
<code>c_format_xml</code>	<code>constant t_format</code>	<code>:= 'XML';</code>
<code>c_format_pxml</code>	<code>constant t_format</code>	<code>:= 'PXML';</code>
<code>c_format_json</code>	<code>constant t_format</code>	<code>:= 'JSON';</code>
<code>c_format_pjson</code>	<code>constant t_format</code>	<code>:= 'PJSON';</code>

Alignment Constants

Constants used in the `ADD_COLUMN`, `ADD_COLUMN_GROUP`, and `GET_PRINT_CONFIG` methods.

<code>c_align_start</code>	<code>constant t_alignment</code>	<code>:= 'LEFT';</code>
<code>c_align_center</code>	<code>constant t_alignment</code>	<code>:= 'CENTER';</code>
<code>c_align_end</code>	<code>constant t_alignment</code>	<code>:= 'RIGHT';</code>

Content Disposition Constants

Constants used in the DOWNLOAD procedure.

```
c_attachment           constant t_content_disposition  :=
'attachment';
c_inline              constant t_content_disposition  := 'inline';
```

Size Unit Constants

Constants used in the GET_PRINT_CONFIG function.

```
c_unit_inches          constant t_unit                  := 'INCHES';
c_unit_millimeters    constant t_unit                  := 'MILLIMETERS';
c_unit_centimeters    constant t_unit                  := 'CENTIMETERS';
c_unit_points          constant t_unit                  := 'POINTS';
```

Predefined Size Constants

Constants used in the GET_PRINT_CONFIG function.

```
c_size_letter          constant t_size                 := 'LETTER';
c_size_legal            constant t_size                 := 'LEGAL';
c_size_tabloid          constant t_size                 := 'TABLOID';
c_size_A4               constant t_size                 := 'A4';
c_size_A3               constant t_size                 := 'A3';
c_size_custom           constant t_size                 := 'CUSTOM';
```

Column Width Unit Constants

Constants used in the GET_PRINT_CONFIG function.

```
c_width_unit_percentage constant t_width_unit        :=
'PERCENTAGE';
c_width_unit_points      constant t_width_unit        := 'POINTS';
c_width_unit_pixels      constant t_width_unit        := 'PIXELS';
```

Page Orientation Constants

Constants used in the GET_PRINT_CONFIG function.

```
c_orientation_portrait   constant t_orientation       := 'VERTICAL';
c_orientation_landscape   constant t_orientation       := 'HORIZONTAL';
```

Font Family Constants

Constants used in the GET_PRINT_CONFIG function.

```
c_font_family_helvetica   constant t_font_family      :=
'Helvetica';
```

c_font_family_times	constant t_font_family	:= 'Times';
c_font_family_courier	constant t_font_family	:= 'Courier';

Font Weight Constants

Constants used in the GET_PRINT_CONFIG function.

c_font_weight_normal	constant t_font_weight	:= 'normal';
c_font_weight_bold	constant t_font_weight	:= 'bold';

20.2 Data Types

The APEX_DATA_EXPORT package uses the following data types.

Generic

subtype t_alignment	is varchar2(255);
subtype t_label	is varchar2(255);
subtype t_color	is varchar2(4000);
subtype t_format	is varchar2(20);
subtype t_content_disposition	is varchar2(30);
subtype t_unit	is varchar2(4000);
subtype t_size	is varchar2(4000);
subtype t_width_unit	is varchar2(255);
subtype t_orientation	is varchar2(4000);
subtype t_font_family	is varchar2(4000);
subtype t_font_weight	is varchar2(4000);

Resulting Object of an Export

```
type t_export is record (
    file_name          varchar2(32767),
    format              t_format,
    mime_type          varchar2(32767),
    as_clob             boolean,
    content_blob        blob,
    content_clob        blob );
```

Column Groups

```
type t_column_group is record (
    name                varchar2(255),
    alignment            t_alignment,
    parent_group_idx     pls_integer );

type t_column_groups      is table of t_column_group index by pls_integer;
```

Columns

```
type t_column is record (
    name                apex_exec.t_column_name,
    heading              varchar2(255),
```

```
format_mask          varchar2(4000),
heading_alignment   t_alignment,
value_alignment     t_alignment,
width               number,
is_column_break    boolean,
is_frozen           boolean,
column_group_idx   pls_integer );  
  
type t_columns          is table of t_column      index by pls_integer;
```

Highlights

```
type t_highlight is record (
  id                  number,
  name                varchar2(4000),
  value_column        apex_exec.t_column_name,
  display_column      apex_exec.t_column_name,
  text_color          t_color,
  background_color    t_color );  
  
type t_highlights       is table of t_highlight  index by pls_integer;
```

Aggregates

```
type t_aggregate is record (
  label               t_label,
  format_mask         varchar2(4000),
  display_column      apex_exec.t_column_name,
  value_column        apex_exec.t_column_name,
  overall_label       t_label,
  overall_value_column apex_exec.t_column_name );  
  
type t_aggregates      is table of t_aggregate index by pls_integer;
```

Print Config

```
type t_print_config is record (
  units                t_unit,
  paper_size           t_size,
  width_units          t_width_unit,
  width                number,
  height               number,
  orientation          t_orientation,
  page_header          varchar2(4000),
  page_header_font_color t_color,
  page_header_font_family t_font_family,
  page_header_font_weight t_font_weight,
  page_header_font_size varchar2(4000),
  page_header_alignment t_alignment,
  page_footer          varchar2(4000),
  page_footer_font_color t_color,
  page_footer_font_family t_font_family,
  page_footer_font_weight t_font_weight,
  page_footer_font_size varchar2(4000),
```

```

page_footer_alignment      t_alignment,
header_bg_color            t_color,
header_font_color          t_color,
header_font_family         t_font_family,
header_font_weight         t_font_weight,
header_font_size            varchar2(4000),
body_bg_color               t_color,
body_font_color             t_color,
body_font_family            t_font_family,
body_font_weight            t_font_weight,
body_font_size               varchar2(4000),
border_width                  number,
border_color                 t_color );

```

20.3 ADD_AGGREGATE Procedure

This procedure adds an aggregate to the aggregate collection. Aggregate collections can be passed to the `EXPORT` calls in order to add an aggregate row. This procedure can be used in combination with control breaks or standalone for overall aggregates.

If an empty aggregate collection (or no aggregate collection) is passed, no aggregate rows render in the export.

This procedure requires an aggregate column. Value is the current aggregate total (for control breaks) or the overall total.

Syntax

```

PROCEDURE ADD_AGGREGATE(
    p_aggregates           IN OUT NOCOPY t_aggregates,
    p_label                 IN          t_label,
    p_format_mask           IN          VARCHAR2          DEFAULT
    NULL,
    p_display_column        IN          apex_exec.t_column_name,
    p_value_column          IN          apex_exec.t_column_name,
    p_overall_label         IN          t_label          DEFAULT
    NULL,
    p_overall_value_column  IN          apex_exec.t_column_name  DEFAULT
    NULL );

```

Parameters

Parameter	Description
p_aggregates	Aggregate collection.
p_label	Aggregate label.
p_format_mask	Format mask to apply on the aggregate value.
p_display_column	Name of the column where to display the aggregate.
p_value_column	Name of the column which contains the value of the aggregate.
p_overall_label	Overall label.
p_overall_value_column	Name of the column which contains the value of the overall aggregate.

Examples

```

DECLARE
    l_aggregates    apex_data_export.t_aggregates;
    l_columns       apex_data_export.t_columns;
    l_context        apex_exec.t_context;
    l_export         apex_data_export.t_export;
BEGIN
    apex_data_export.add_aggregate(
        p_aggregates          => l_aggregates,
        p_label                => 'Sum',
        p_format_mask          => 'FML999G999G999G999G990D00',
        p_display_column       => 'SAL',
        p_value_column         => 'AGGREGATE1',
        p_overall_label        => 'Total sum',
        p_overall_value_column => 'OVERALL1' );

    apex_data_export.add_column( p_columns => l_columns, p_name => 'DEPTNO',
p_is_column_break => true );
    apex_data_export.add_column( p_columns => l_columns, p_name => 'EMPNO');
    apex_data_export.add_column( p_columns => l_columns, p_name => 'ENAME');
    apex_data_export.add_column( p_columns => l_columns, p_name => 'SAL');

    l_context := apex_exec.open_query_context(
        p_location      => apex_exec.c_location_local_db,
        p_sql_query     => 'select deptno,
                           empno,
                           ename,
                           sal,
                           sum( sal ) over ( partition by deptno ) as
AGGREGATE1,
                           sum( sal ) over ( ) as OVERALL1
        FROM emp
        order by deptno' );

    l_export := apex_data_export.export (
        p_context      => l_context,
        p_format       => apex_data_export.c_format_pdf,
        p_columns      => l_columns,
        p_aggregates   => l_aggregates );

    apex_exec.close( l_context );

    apex_data_export.download( p_export => l_export );

EXCEPTION
    WHEN others THEN
        apex_exec.close( l_context );
        raise;
END;

```

20.4 ADD_COLUMN Procedure

This procedure adds a column to the column collection. Column collections can be passed to the `EXPORT` calls in order to return only a subset of the columns in the export. If an empty column collection (or no column collection) passes, all columns defined in the Query Context are added to the export.

Syntax

```
PROCEDURE ADD_COLUMN (
    p_columns          IN OUT NOCOPY t_columns,
    p_name              IN           apex_exec.t_column_name,
    p_heading            IN           VARCHAR2                      DEFAULT NULL,
    p_format_mask        IN           VARCHAR2                      DEFAULT NULL,
    p_heading_alignment  IN           t_alignment                  DEFAULT NULL,
    p_value_alignment   IN           t_alignment                  DEFAULT NULL,
    p_width              IN           NUMBER                      DEFAULT NULL,
    p_is_column_break   IN           BOOLEAN                     DEFAULT FALSE,
    p_is_frozen          IN           BOOLEAN                     DEFAULT FALSE,
    p_column_group_idx  IN           PLS_INTEGER                DEFAULT
NULL );
```

Parameters

Parameter	Description
p_columns	Column collection.
p_name	Column name.
p_heading	Column heading text.
p_format_mask	Format mask to apply. Useful for XLSX exports where native datatypes are used.
p_heading_alignment	Column heading alignment. Valid values are: LEFT, CENTER, RIGHT.
p_value_alignment	Column value alignment. Valid values are: LEFT, CENTER, RIGHT.
p_width	PDF only. The column width. By default the units are as percentage. The units can be modified by updating the width_units of the print config.
p_is_column_break	Whether to use this column for control breaks
p_is_frozen	XLSX only. Whether the column is frozen.
p_column_group_idx	The index of a column group. If used, this column will part of the column group.

Examples

```
DECLARE
    l_context      apex_exec.t_context;
    l_export        apex_data_export.t_export;
    l_columns       apex_data_export.t_columns;
```

```

BEGIN
    l_context := apex_exec.open_query_context(
        p_location      => apex_exec.c_location_local_db,
        p_sql_query     => 'select * from emp' );

    apex_data_export.add_column(
        p_columns          => l_columns,
        p_name              => 'ENAME',
        p_heading           => 'Name' );

    apex_data_export.add_column(
        p_columns          => l_columns,
        p_name              => 'JOB',
        p_heading           => 'Job' );

    apex_data_export.add_column(
        p_columns          => l_columns,
        p_name              => 'SAL',
        p_heading           => 'Salary',
        p_format_mask       => 'FML999G999G999G999G990D00' );

    l_export := apex_data_export.export (
        p_context          => l_context,
        p_format            => apex_data_export.c_format_html,
        p_columns           => l_columns,
        p_file_name         => 'employees' );

    apex_exec.close( l_context );

    apex_data_export.download( p_export => l_export );

EXCEPTION
    WHEN others THEN
        apex_exec.close( l_context );
        raise;
END;

```

20.5 ADD_COLUMN_GROUP Procedure

This procedure adds a column group to the column group collection. Column group collections can be passed to the EXPORT calls in order to group columns using an extra header row. If an empty column group collection (or no column group collection) passes, no column groups are added to the export. You can create multiple column group levels.

Syntax

```

APEX_DATA_EXPORT.ADD_COLUMN_GROUP (
    p_column_groups    IN OUT NOCOPY    t_column_groups,
    p_idx               OUT             PLS_INTEGER,
    p_name              IN              VARCHAR2,
    p_alignment         IN              t_alignment      DEFAULT
c_align_center,
    p_parent_group_idx IN              PLS_INTEGER      DEFAULT NULL )

```

Parameters

Parameter	Description
p_column_groups	Column group collection.
p_idx	The generated index in the columns collection.
p_name	Column group name.
p_alignment	Column group alignment. Valid values are: LEFT, CENTER (default), RIGHT.
p_parent_group_idx	The index of a parent column group.

Example

```

DECLARE
    l_context          apex_exec.t_context;
    l_export           apex_data_export.t_export;
    l_column_groups   apex_data_export.t_column_groups;
    l_columns          apex_data_export.t_columns;

    -- Column group indexes
    l_identity_idx     pls_integer;
    l_compensation_idx pls_integer;
BEGIN

    l_context := apex_exec.open_query_context(
        p_location      => apex_exec.c_location_local_db,
        p_sql_query     => 'select * from emp');

    -- Define column groups
    apex_data_export.add_column_group(
        p_column_groups  => l_column_groups,
        p_idx            => l_identity_idx,
        p_name           => 'Identity');

    apex_data_export.add_column_group(
        p_column_groups  => l_column_groups,
        p_idx            => l_compensation_idx,
        p_name           => 'Compensation');

    -- Define columns
    apex_data_export.add_column(
        p_columns         => l_columns,
        p_name            => 'ENAME',
        p_heading         => 'Name',
        p_column_group_idx => l_identity_idx);

    apex_data_export.add_column(
        p_columns         => l_columns,
        p_name            => 'JOB',
        p_heading         => 'Job',
        p_column_group_idx => l_identity_idx);

    apex_data_export.add_column(

```

```

        p_columns          => l_columns,
        p_name             => 'SAL',
        p_heading          => 'Salary',
        p_column_group_idx => l_compensation_idx );

apex_data_export.add_column(
        p_columns          => l_columns,
        p_name             => 'COMM',
        p_heading          => 'Commission',
        p_column_group_idx => l_compensation_idx );

l_export := apex_data_export.export (
        p_context          => l_context,
        p_format            => apex_data_export.c_format_html,
        p_columns           => l_columns,
        p_column_groups    => l_column_groups,
        p_file_name         => 'employees' );

apex_exec.close( l_context );

apex_data_export.download( p_export => l_export );

EXCEPTION
    when others THEN
        apex_exec.close( l_context );
        raise;
END;

```

20.6 ADD_HIGHLIGHT Procedure

This procedure adds a highlight to the highlight collection. Highlight collections can be passed to the EXPORT calls in order to highlight a row or a column in a row. If no highlight collection (or an empty highlight collection) is passed, no highlights render in the export.

This procedure requires a highlight column. The value is the ID when highlights should be applied, else NULL.

Syntax

```

PROCEDURE ADD_HIGHLIGHT (
    p_highlights      IN OUT NOCOPY t_highlights,
    p_id              IN          pls_integer,
    p_value_column    IN          apex_exec.t_column_name,
    p_display_column  IN          apex_exec.t_column_name DEFAULT NULL,
    p_text_color       IN          t_color                DEFAULT NULL,
    p_background_color IN          t_color                DEFAULT NULL );

```

Parameters

Parameter	Description
p_highlights	Highlight collection.
p_id	ID of the highlight.

Parameter	Description
p_value_column	Name of the column where to check for the highlight ID.
p_display_column	Name of the column where to display the highlight. Leave empty for row highlights.
p_text_color	Hex color code of the text (#FF0000).
p_background_color	Hex color code of the background (#FF0000).

Examples

```

DECLARE
    l_highlights      apex_data_export.t_highlights;
    l_context         apex_exec.t_context;
    l_export          apex_data_export.t_export;
BEGIN
    apex_data_export.add_highlight(
        p_highlights      => l_highlights,
        p_id              => 1,
        p_value_column    => 'HIGHLIGHT1',
        p_display_column  => 'SAL',
        p_text_color      => '#FF0000' );

    l_context := apex_exec.open_query_context(
        p_location      => apex_exec.c_location_local_db,
        p_sql_query     => 'select empno,
                                ename,
                                sal,
                                case when sal >= 3000 then 1 end as HIGHLIGHT1
                         from emp' );

    l_export := apex_data_export.export (
        p_context      => l_context,
        p_format       => apex_data_export.c_format_pdf,
        p_highlights   => l_highlights );

    apex_exec.close( l_context );

    apex_data_export.download( p_export => l_export );

EXCEPTION
    when others THEN
        apex_exec.close( l_context );
        raise;
END;

```

20.7 DOWNLOAD Procedure

This procedure downloads the data export by calling `APEX_APPLICATION.STOP_APEX_ENGINE`.

Syntax

```
APEX_DATA_EXPORT.DOWNLOAD (
    p_export          IN OUT NOCOPY t_export,
    p_content_disposition IN t_content_disposition DEFAULT c_attachment,
    p_add_file_extension   IN BOOLEAN      DEFAULT TRUE,
    p_stop_apex_engine     IN BOOLEAN      DEFAULT TRUE )
```

Parameters

Parameter	Description
p_export	The result object of an export.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment" or "inline").
p_add_file_extension	Whether Oracle APEX adds the file extension to the filename based on the export format.
p_stop_apex_engine	Whether to call APEX_APPLICATION.STOP_APEX_ENGINE.

Examples

```
DECLARE
    l_context apex_exec.t_context;
    l_export apex_data_export.t_export;
BEGIN
    l_context := apex_exec.open_query_context(
        p_location    => apex_exec.c_location_local_db,
        p_sql_query   => 'select * from emp' );

    l_export := apex_data_export.export (
        p_context    => l_context,
        p_format     => apex_data_export.c_format_csv,
        p_file_name  => 'employees' );

    apex_exec.close( l_context );
    apex_data_export.download( p_export => l_export );

EXCEPTION
    when others THEN
        apex_exec.close( l_context );
        raise;
END;
```

20.8 EXPORT Function

This function exports the query context in the specified format.

Syntax

```
FUNCTION EXPORT (
    p_context          IN apex_exec.t_context,
```

```

p_format                      IN t_format,
p_as_clob                     IN BOOLEAN          DEFAULT false,
p_columns                      IN t_columns         DEFAULT c_empty_columns,
p_column_groups                IN t_column_groups  DEFAULT c_empty_column_groups,
p_aggregates                   IN t_aggregates    DEFAULT c_empty_aggregates,
p_highlights                    IN t_highlights     DEFAULT c_empty_highlights,
--
p_file_name                    IN VARCHAR2        DEFAULT NULL,
p_print_config                 IN t_print_config   DEFAULT c_empty_print_config,
p_page_header                  IN VARCHAR2        DEFAULT NULL,
p_page_footer                  IN VARCHAR2        DEFAULT NULL,
p_supplemental_text            IN VARCHAR2        DEFAULT NULL,
--
p_csv_enclosed_by              IN VARCHAR2        DEFAULT NULL,
p_csv_separator                IN VARCHAR2        DEFAULT NULL,
--
p_pdf_accessible               IN BOOLEAN         DEFAULT NULL,
--
p_xml_includeDeclaration       IN BOOLEAN         DEFAULT false )
RETURN t_export

```

Parameters

Parameter	Description
p_context	Context object from the EXEC infrastructure.
p_format	Export format. Valid values are: XLSX, PDF, HTML, CSV, XML and JSON.
p_as_clob	Exports as a CLOB instead of BLOB (default FALSE).
p_columns	Collection of column attributes beginning with column breaks, then in the order of display.
p_column_groups	Collection of column group attributes in the order of levels and display.
p_aggregates	Collection of report aggregates.
p_highlights	Collection of report highlights.
p_file_name	Defines the filename of the export.
p_print_config	Used for EXCEL and PDF to set the print attributes.
p_page_header	Text to appear in the header section of the document. Overrides the page header from p_print_config.
p_page_footer	Text to appear in the footer section of the document. Overrides the page footer from p_print_config.
p_supplemental_text	Text at the top of all download formats.
p_csv_enclosed_by	Used for CSV to enclose the output.
p_csv_separator	Used for CSV to separate the column values.
p_pdf_accessible	Used for PDF to create an accessible PDF.
p_xml_include_declaration	Used for XML to generate the XML declaration as the first line.

Returns

This function returns: the export file as object which includes the contents, MIME type, and file name.

Examples

```

DECLARE
    l_context apex_exec.t_context;
    l_export  apex_data_export.t_export;
BEGIN
    l_context := apex_exec.open_query_context(
        p_location    => apex_exec.c_location_local_db,
        p_sql_query   => 'select * from emp' );

    l_export := apex_data_export.export (
        p_context     => l_context,
        p_format      => apex_data_export.c_format_pdf );

    apex_exec.close( l_context );

    apex_data_export.download( p_export => l_export );

EXCEPTION
    when others THEN
        apex_exec.close( l_context );
        raise;
END;

```

20.9 GET_PRINT_CONFIG Procedure

This function prepares the print config to style the data export.

- The colors are specified using hexadecimal (hex) notation, RGB color codes, or HTML color names.
- The alignment options include: Left, Center, Right
- The font family options include: Helvetica, Times, Courier
- The font weight options include: Normal, Bold

Syntax

```

FUNCTION GET_PRINT_CONFIG(
    p_units                      IN t_unit          DEFAULT c_unit_inches,
    p_paper_size                  IN t_size          DEFAULT c_size_letter,
    p_width_units                IN t_width_unit  DEFAULT
c_width_unit_percentage,
    p_width                     IN NUMBER         DEFAULT 11,
    p_height                     IN NUMBER         DEFAULT 8.5,
    p_orientation                IN t_orientation DEFAULT
c_orientation_landscape,
    --
    p_page_header                IN VARCHAR2       DEFAULT NULL,
    p_page_header_font_color    IN t_color         DEFAULT '#000000',

```

```

    p_page_header_font_family      IN t_font_family  DEFAULT
c_font_family_helvetica,
    p_page_header_font_weight    IN t_font_weight  DEFAULT
c_font_weight_normal,
    p_page_header_font_size      IN NUMBER        DEFAULT 12,
    p_page_header_alignment     IN t_alignment   DEFAULT c_align_center,
    --
    p_page_footer                 IN VARCHAR2     DEFAULT NULL,
    p_page_footer_font_color    IN t_color       DEFAULT '#000000',
    p_page_footer_font_family   IN t_font_family DEFAULT
c_font_family_helvetica,
    p_page_footer_font_weight    IN t_font_weight DEFAULT
c_font_weight_normal,
    p_page_footer_font_size      IN NUMBER        DEFAULT 12,
    p_page_footer_alignment     IN t_alignment   DEFAULT c_align_center,
    --
    p_header_bg_color            IN t_color       DEFAULT '#EEEEEE',
    p_header_font_color          IN t_color       DEFAULT '#000000',
    p_header_font_family         IN t_font_family DEFAULT
c_font_family_helvetica,
    p_header_font_weight         IN t_font_weight DEFAULT
c_font_weight_bold,
    p_header_font_size           IN NUMBER        DEFAULT 10,
    --
    p_body_bg_color              IN t_color       DEFAULT '#FFFFFF',
    p_body_font_color            IN t_color       DEFAULT '#000000',
    p_body_font_family           IN t_font_family DEFAULT
c_font_family_helvetica,
    p_body_font_weight           IN t_font_weight DEFAULT
c_font_weight_normal,
    p_body_font_size             IN NUMBER        DEFAULT 10,
    --
    p_border_width               IN NUMBER        DEFAULT .5,
    p_border_color                IN t_color       DEFAULT '#666666' )
RETURN t_print_config;

```

Parameters

Parameter	Description
p_units	Select the units used to specify page width and height. Valid values are: Inches, Millimeters, Centimeters, Points
p_paper_size	PDF only. Select the report page size. To type in your own page width and height, select Custom. Available options include: Letter, Legal, Tabloid, A4, A3, Custom
p_width_units	PDF only. Select the units used to specify column widths. Valid values are: Percentage, Points, Pixels
p_width	PDF only. The width of the page.
p_height	PDF only. The height of the page.

Parameter	Description
p_orientation	The orientation for the page. PDF only. Available options include: Vertical (Portrait), Horizontal (Landscape)
p_page_header	Text to appear in the header section of the document.
p_page_header_font_color	The page header font color.
p_page_header_font_family	The page header font family.
p_page_header_font_weight	The page header font weight.
p_page_header_font_size	The page header font size.
p_page_header_alignment	The page header text alignment.
p_page_footer	Text to appear in the footer section of the document.
p_page_footer_font_color	The page footer font color.
p_page_footer_font_family	The page footer font family.
p_page_footer_font_weight	The page footer font weight.
p_page_footer_font_size	The page footer font size.
p_page_footer_alignment	The page footer text alignment.
p_header_bg_color	The table header background color.
p_header_font_color	The table header font color.
p_header_font_family	The table header font family.
p_header_font_weight	The table header font weight.
p_header_font_size	The table header font size.
p_body_bg_color	The table body background color.
p_body_font_color	The table body font color.
p_body_font_family	The table body font family.
p_body_font_weight	The table body font weight.
p_body_font_size	The table body font size.
p_border_width	The width of the borders.
p_border_color	The color of the borders.

Returns

The print config to style the data export.

Example

```

DECLARE
    l_context      apex_exec.t_context;
    l_print_config apex_data_export.t_print_config;
    l_export       apex_data_export.t_export;
BEGIN
    l_context := apex_exec.open_query_context(
        p_location    => apex_exec.c_location_local_db,
        p_sql_query   => 'select * from emp' );

    l_print_config := apex_data_export.get_print_config(
        p_orientation  => apex_data_export.c_orientation_portrait,
        p_border_width => 2 );

```

```
l_export := apex_data_export.export (
    p_context      => l_context,
    p_print_config => l_print_config,
    p_format        => apex_data_export.c_format_pdf );

apex_exec.close( l_context );

apex_data_export.download( p_export => l_export );

EXCEPTION
    when others THEN
        apex_exec.close( l_context );
        raise;
END;
```

21

APEX_DATA_INSTALL

This package contains the API for data migration in Oracle APEX.

- [LOAD_SUPPORTING_OBJECT_DATA Procedure](#)

21.1 LOAD_SUPPORTING_OBJECT_DATA Procedure

This procedure loads the supporting object data and installs the data in the specified application.

Syntax

```
APEX_DATA_INSTALL.LOAD_SUPPORTING_OBJECT_DATA (
    p_table_name          IN VARCHAR2,
    p_delete_after_install IN BOOLEAN,
    p_app_id              IN NUMBER DEFAULT NULL );
```

Parameters

Parameter	Description
p_table_name	Name of the table where the data will be deposited.
p_delete_after_install	Indicates if files are removed after installing supporting objects. Default TRUE.
p_app_id	APEX application ID of the application that contains the static files associated with a data migration export. This can be used from SQL workshop outside the context of installing supporting objects, enabling a developer to reinstall migrated data without reinstalling all supporting objects.

Example

```
DECLARE
    l_table_name      varchar2(400);
BEGIN
    apex_data_install.load_supporting_object_data(
        p_table_name      => l_table_name,
        p_delete_after_install => true);
END;
```

APEX_DATA_PARSER

This package contains the implementation for the file parser in Oracle 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 APEX_T_PARSER_ROW type. The parser supports up to 300 columns.

- [Global Constants](#)
- [Data Types](#)
- [ASSERT_FILE_TYPE Function](#)
- [DISCOVER Function](#)
- [GET_COLUMNS Function](#)
- [GET_FILE_PROFILE Function](#)
- [GET_FILE_TYPE Function](#)
- [GET_XLSX_WORKSHEETS Function](#)
- [JSON_TO_PROFILE Function](#)
- [PARSE Function](#)
- [SET_PARSER_FLAGS Procedure](#)

22.1 Global Constants

The APEX_DATA_PARSER package uses the following constants.

```
subtype t_file_type is pls_integer range 1..5;
c_file_type_xlsx      constant t_file_type := 1; -- File Type Constant
c_file_type_csv       constant t_file_type := 2; -- File Type Constant
c_file_type_xml       constant t_file_type := 3; -- File Type Constant
c_file_type_json      constant t_file_type := 4; -- File Type Constant
c_file_type_ics       constant t_file_type := 5; -- File Type Constant
```

22.2 Data Types

The APEX_DATA_PARSER package uses the following data types.

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),
csv_enclosed        VARCHAR2(4),
null_if             VARCHAR2(20),
force_trim_whitespace BOOLEAN,
skip_rows           PLS_INTEGER,
file_columns         t_file_columns
);

type t_file_columns is table of t_file_column index by pls_integer;

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,
    is_json           BOOLEAN,
    selector          VARCHAR2(32767),
    decimal_char      VARCHAR2(4),
    group_char        VARCHAR2(4),
    format_mask       VARCHAR2(128),
    clob_column       PLS_INTEGER
);

```

22.3 ASSERT_FILE_TYPE Function

This function checks whether the extension of the file name matches the specified file type.

Syntax

```
APEX_DATA_PARSER ASSERT_FILE_TYPE (
    p_file_name IN VARCHAR2,
    p_file_type IN t_file_type )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_file_name	File name to get the file type.
p_file_type	File type as t_file_type.

Returns

TRUE, if the file name matches the specified extension. FALSE otherwise.

Example

The following example checks if the passed-in file name is the CSV file type.

```
DECLARE
    is_valid_file_type boolean;
BEGIN
    is_valid_file_type := apex_data_parser.assert_file_type(
```

```

    p_file_name => 'test.csv',
    p_file_type => apex_data_parser.c_file_type_csv );
END;

```

22.4 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

```

APEX_DATA_PARSER.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_skip_rows        IN PLS_INTEGER DEFAULT NULL,
    --
    p_nullif           IN VARCHAR2      DEFAULT NULL,
    p_force_trim_whitespace IN VARCHAR2 DEFAULT 'Y',
    --
    p_file_charset      IN VARCHAR2    DEFAULT 'AL32UTF8',
    p_max_rows          IN NUMBER       DEFAULT 200,
    --
    p_xml_namespaces   IN VARCHAR2      DEFAULT NULL )
RETURN CLOB;

```

Parameter

Parameter	Description
p_content	The file content to be parsed as a BLOB.
p_file_name	The name of the file used to derive the file type.
p_decimal_char	Use this decimal character when trying to detect NUMBER data types. If not specified, the procedure will auto-detect the decimal character.
p_xlsx_sheet_name	For XLSX workbooks. The name of the worksheet to parse. If omitted, the function uses the first worksheet found.

Parameter	Description
p_row_selector	Row selector to use for XML or JSON files. Pointer to the array/list of rows within the JSON or XML file. If omitted, the function will: <ul style="list-style-type: none"> For XML files: Use /*/* (first tag under the root tag) as the row selector. For JSON files: Look for a JSON array and use the first array found.
p_csv_row_delimiter	Override the default row delimiter for CSV parsing.
p_csv_col_delimiter	Use a specific CSV column delimiter. If omitted, the function detects the column delimiter based on the first row contents.
p_csv_enclosed	Override the default enclosure character for CSV parsing.
p_skip_rows	The amount of rows to skip before parsing. For XML and JSON parsing to get the same behavior as for CSV and XLSX parsing.
p_nullif	Similar to SQL NULLIF function: If the column has this value, return NULL.
p_force_trim_whitespace	Whether to force trim enquoted whitespace from parsed values.
p_file_charset	File encoding, if not UTF-8 (AL32UTF8).
p_max_rows	Stop discovery after P_MAX_ROWS rows have been processed.
p_xml_namespaces	XML namespaces to use when parsing XML files.

Returns

CLOB containing the file profile in JSON format.

Example

```
select apex_data_parser.discover(
    p_content => {BLOB containing XLSX file},
    p_file_name=>'large.xlsx' ) as profile_json
from dual;
```

PROFILE_JSON

```
{
  "file-encoding" : "AL32UTF8",
  "single-row" : false,
  "file-type" : 1,
  "parsed-rows" : 2189,
  "columns" : [
    {
      "name" : "C0",
      "format-mask" : "",
      "selector" : "",
      "data-type" : 2
    },
    {
      "selector" : "",
      "format-mask" : "",
      "data-type" : 1,
      "name" : "FIRST_NAME"
    }
  ]
}
```

```
{  
    "name" : "LAST_NAME",  
    "format-mask" : "",  
    "selector" : "",  
    "data-type" : 1  
,  
:  
  
{  
    "name" : "DATE_",  
    "format-mask" : "DD\\"/\\"MM\\"/\\"YYYY",  
    "data-type" : 3,  
    "selector" : ""  
,  
{  
    "format-mask" : "",  
    "selector" : "",  
    "data-type" : 2,  
    "name" : "ID"  
}  
,  
]  
,  
"row-selector" : "",  
"headings-in-first-row" : true,  
"xlsx-worksheet" : "sheet1.xml",  
"csv-delimiter" : ""  
}
```

22.5 GET_COLUMNS Function

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

Syntax

```
APEX_DATA_PARSER.GET_COLUMNS (  
    p_profile           IN CLOB )  
    RETURN apex_t_parser_columns PIPELINED;
```

Parameter

Parameter	Description
p_profile	File profile to be used for parsing. The file profile might have been computed in a previous <code>parse()</code> or <code>discover()</code> invocation.

Returns

Returns profile column information as rows of `apex_t_parser_columns`.

Example

The following example uses `discover()` to compute a file profile and then `get_columns()` to return the list of columns.

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

COLUMN_POSITION	COLUMN_NAME	DATA_TYPE	FORMAT_MASK
1	C0	NUMBER	
2	FIRST_NAME	VARCHAR2	
3	LAST_NAME	VARCHAR2	
4	GENDER	VARCHAR2	
5	COUNTRY	VARCHAR2	
6	AGE	NUMBER	
7	DATE_	DATE	DD"/"MM"/"YYYY
8	ID	NUMBER	

22.6 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 metadata 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. The `GET_COLUMNS` function can be used to display file profile information in a structured way.
3. Process the data.

Syntax

```
APEX_DATA_PARSER.GET_FILE_PROFILE RETURN CLOB;
```

Parameter

None.

Returns

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

Example

```

        "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" : ""
    }
],
"row-selector" : ""
}

```

22.7 GET_FILE_TYPE Function

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

Syntax

```

APEX_DATA_PARSER.GET_FILE_TYPE (
    p_file_name IN VARCHAR2 )
RETURN t_file_type;

```

Parameter

Parameter	Description
p_file_name	File name to get the file type.

Returns

Returns the file type as t_file_type.

Example

```
DECLARE
    l_file_type apex_data_parser.t_file_type;
BEGIN
    l_file_type := apex_data_parser.get_file_type( 'test.xlsx' );
END;
```

22.8 GET_XLSX_WORKSHEETS Function

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

Syntax

```
APEX_DATA_PARSER.GET_XLSX_WORKSHEETS (
    p_content    IN BLOB )
RETURN apex_t_parser_worksheets;
```

Parameter

Parameter	Description
p_content	XLSX worksheet as a BLOB.

Returns

Returns table with worksheet information.

Example

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

SHEET_SEQUENCE SHEET_DISPLAY_NAME SHEET_FILE_NAME SHEET_PATH
1             Sheet1           sheet1.xml      worksheets/sheet1.xml
```

22.9 JSON_TO_PROFILE Function

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

Syntax

```
APEX_DATA_PARSER.JSON_TO_PROFILE (
    p_json    IN CLOB )
    RETURN t_file_profile;
```

Parameter

Parameter	Description
<code>p_json</code>	The data profile in JSON format.

Returns

Returns the the file profile in JSON format.

Example

```
DECLARE
    l_profile t_file_profile;
BEGIN
    l_profile := apex_data_parser.json_to_profile( '{"file-type": "csv-delimiter": "", ... }' );
END;
```

22.10 PARSE Function

This function enables you to parse XML, XLSX, CSV, or JSON files and returns a generic table of the following structure:

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

Values are generally returned in `VARCHAR2` format. A returned table row can have a maximum of 300 columns. The maximum length for a `VARCHAR2` table column is 4000 bytes; there is no line length maximum. 20 out of the 300 supported columns can be handled as a `CLOB`.

File parsing happens on-the-fly as this function is invoked. Data does not write to a collection nor to a temporary table.

About Parsing File Profiles

If the `p_file_profile` parameter is not passed, the function computes a file profile with column information during parsing.

If `p_detect_data_types` is passed as `Y` (default), the function also detects column data types during parsing. Retrieve the computed file profile using `GET_FILE_PROFILE` after the function finishes:

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:**

XLSX parsing occurs in phases:

1. First, APEX_ZIP extracts individual XML files from the XLSX archive.
2. Then, the XMLTABLE SQL function parses the actual XLSX.

About CLOB Support

Starting with APEX release 19.2, this package supports string values larger than 4,000 bytes. 20 out of the 300 supported columns can be handled as a CLOB. The level of CLOB support depends upon the file type being parsed.

CSV and XLSX

- CLOB values are supported up to 32K.
- CLOB columns can be detected during discovery.
- When the data profile is discovered, values below 4000 bytes are normally returned as COLNNN. CLOB values are returned in the CLOBNN column and the first 1000 characters are returned as COLNNN. If a data profile is passed in and that has CLOB column defined, all values are returned in the CLOBNN column only.

XML

- CLOB values with more than 32K are supported.
- CLOB columns can be detected during discovery.
- When the data profile is discovered, values below 4000 bytes are normally returned as COLNNN. CLOB values are returned in the CLOBNN column and the first 1000 characters are returned as COLNNN. If a data profile is passed in and that has CLOB column defined, all values are returned in the CLOBNN column only.

JSON

- CLOB values with more than 32K are supported.
- CLOB columns are **not** detected during discovery; CLOB support is only active if a file profile containing CLOB column is passed in as the p_file_profile parameter.
- Since JSON_TABLE does not support CLOBS on 12c databases, the parser uses XMLTYPE-based processing if a file profile with CLOB columns is passed in. Processing will be significantly slower.

About Large CSV Files

If the BLOB passed to APEX_DATA_PARSER.PARSE is less than 50 MB, Oracle APEX copies the BLOB to an *internal, cached* temporary LOB. Thus all CSV parsing is done in memory. For larger BLOBs, APEX does CSV parsing on the original BLOB locator. If it is selected from a

table, CSV parsing can happen on disk but might be significantly slower. Note that a performance degradation may occur when parsed CSV files grow beyond 50 MB.

However, developers can also use the `DBMS_LOB.CREATETEMPORARY` (passing `CACHE => TRUE`) and `DBMS_LOB.COPY` procedures in order to explicitly create a cached temporary LOB, even for a larger file. Instead of the original BLOB, the cached temporary LOB can be passed to `APEX_DATA_PARSER.PARSE`. This approach also enables in-memory parsing for files larger than 50 MB.

See Also:

`CREATETEMPORARY` Procedures and `COPY` Procedures in *Oracle Database PL/SQL Packages and Types Reference*.

Syntax

```
APEX_DATA_PARSER.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 '',
    --
    p_skip_rows                     IN PLS_INTEGER   DEFAULT NULL,
    p_add_headers_row               IN VARCHAR2      DEFAULT 'N',
    --
    p_nullif                        IN VARCHAR2      DEFAULT NULL,
    p_force_trim_whitespace         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,
    p_xml_namespaces                IN VARCHAR2      DEFAULT NULL,
    --
    p_fix_excel_precision           IN VARCHAR2      DEFAULT 'N' )
RETURN apex_t_parser_table pipelined;
```

Parameters

Parameter	Description
p_content	The file content to be parsed as a BLOB.
p_file_name	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.
p_file_type	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.
p_file_profile	File profile to be used for parsing. The file profile might have been computed in a previous parse() or discover() invocation. If passed in again, the function skips some profile detection logic and use the passed in profile in order to improve performance.
p_detect_data_types	Whether to detect data types (NUMBER, DATE, TIMESTAMP) during parsing. If Y (default), the function computes the file profile and also add data type information to it. If N, all columns are VARCHAR2. When no data types are detected, all columns are reported as VARCHAR2 (4000).
p_decimal_char	Use this decimal character when trying to detect NUMBER data types. If not specified, the procedure will auto-detect the decimal character.
p_xlsx_sheet_name	For XLSX workbooks. The name of the worksheet to parse. If omitted, the function uses the first worksheet found.
p_row_selector	For JSON and XML files. Pointer to the array / list of rows within the JSON or XML file. If omitted, the function will: <ul style="list-style-type: none">• For XML files: Use /*/* (first tag under the root tag) as the row selector.• For JSON files: Look for a JSON array and use the first array found.
p_csv_row_delimiter	Override the default row delimiter for CSV parsing. Limited to one character and defaults to Linefeed (LF). Note that the Linefeed row delimiter also handles "Carriage Return/Linefeed" (CRLF).
p_csv_col_delimiter	Use a specific CSV column delimiter. If omitted, the function will detect the column delimiter based on the first row contents.
p_csv_enclosed	Override the default enclosure character for CSV parsing.
p_skip_rows	Skip the first N rows when parsing.
p_add_headers_row	For XML, JSON: Emit the column headers (tag, attr names) as the first row.
p_nullif	Similar to SQL NULLIF function. If the column has this value, return NULL.
p_force_trim_whitespace	Whether to force trim enquoted whitespace from parsed values.
p_file_charset	Encoding of the file to parse. Defaults to AL32UTF8 if omitted or NULL is explicitly passed in.
p_max_rows	Stop parsing after p_max_rows have been returned.

Parameter	Description
p_return_rows	Amount of rows to return. This is useful when the parser parses 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 <code>rownum < x</code> clause of the SQL query kicks in and stops execution.
p_store_profile_to_collection	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.
p_xml_namespaces	XML namespaces to use when parsing XML files.
p_fix_excel_precision	Whether to round numbers in XLSX files to 15 significant digits. This is useful for XLSX files generated by Microsoft Excel. Excel stores numeric values as floating point numbers with a maximum of 15 significant digits. For calculation results, this can lead to rounding issues, which are fixed using this parameter. See also: Floating-point arithmetic may give inaccurate results in Excel at Microsoft 365 .

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(
    apex_data_parser.parse(
      p_content      => {BLOB containing XLSX spreadsheet},
      p_file_name   => 'test.xlsx',
      p_xlsx_sheet_name => 'sheet1.xml') ) ;
```

	LINE_NUMBER	COL001	COL002	COL003	COL004	COL005
	COL006	COL007	COL008			
		1 0	First Name	Last Name	Gender	Country
Age	Date		Id			
	2 1	Dulce	Abril	Female	United States	
32	15/10/2017	1562				
	3 2	Mara	Hashimoto	Female	Great Britain	
25	16/08/2016	1582				
	4 3	Philip	Gent	Male	France	
36	21/05/2015	2587				
	5 4	Kathleen	Hanner	Female	United States	
25	15/10/2017	3549				
	6 5	Nereida	Magwood	Female	United States	
58	16/08/2016	2468				
	7 6	Gaston	Brumm	Male	United States	
24	21/05/2015	2554				
	8 7	Etta	Hurn	Female	Great Britain	
56	15/10/2017	3598				

```

         9 8      Earlean      Melgar      Female   United States
27    16/08/2016  2456
          10 9      Vincenza     Weiland      Female   United States
40    21/05/2015  6548
          : :       :           :           :           :
          : :       :           :           :           :

select line_number, col001,col002,col003,col004,col005,col006,col007,col008
  from table(
        apex_data_parser.parse(
          p_content      => {BLOB containing JSON file},
          p_file_name    => 'test.json') ) ;

LINE_NUMBER COL001      COL002      COL003
COL004          COL005
-----
1 Feature      1.5      41km E of Cape Yakataga, Alaska
1536513727239 1536514117117
2 Feature      0.21     11km ENE of Aguanga, CA
1536513299520 1536513521231
3 Feature      1.84     5km SSW of Pahala, Hawaii
1536513262940 1536513459610
4 Feature      2.55     9km W of Volcano, Hawaii
1536513100890 1536513446680
5 Feature      1.3      62km ESE of Cape Yakataga, Alaska
1536512917361 1536513322236
6 Feature      1.79     7km SW of Tiptonville, Tennessee
1536512379690 1536512668010
7 Feature      1.9      126km NNW of Arctic Village, Alaska
1536512346186 1536512846567
8 Feature      1.4      105km NW of Arctic Village, Alaska
1536512140162 1536512846334

```

See Also:

- CREATETEMPORARY Procedures in *Oracle Database PL/SQL Packages and Types Reference*
- COPY Procedures in *Oracle Database PL/SQL Packages and Types Reference*
- Floating-point arithmetic may give inaccurate results in Excel at [Microsoft 365](#)

22.11 SET_PARSER_FLAGS Procedure

This procedure sets flags to control parser behavior. Existing flags include:

- CSV_BACKSLASH_ESCAPING - If Y, the parser treats the backslash character as an additional escape for the enclosed character. Defaults to Y for backwards compatibility.

Syntax

```
APEX_DATA_PARSER.SET_PARSER_FLAGS (
    p_name    IN VARCHAR2,
    p_value   IN VARCHAR2 )
```

Parameters

Parameter	Description
p_name	Name of the flag to set.
p_value	Value to set.

APEX_DEBUG

The `APEX_DEBUG` package provides utility functions for managing the debug message log. Specifically, this package provides the necessary APIs to instrument and debug PL/SQL code contained within your Oracle APEX 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 APEX App Builder User's Guide* or by querying the `APEX_DEBUG_MESSAGES` view.

For further information, see the individual API descriptions.

 **Note:**

In APEX release 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 APEX.

- [Constants](#)
- [DISABLE Procedure](#)
- [DISABLE_DBMS_OUTPUT Procedure](#)
- [ENABLE Procedure](#)
- [ENTER Procedure](#)
- [ENABLE_DBMS_OUTPUT Procedure](#)
- [ERROR Procedure](#)
- [GET_LAST_MESSAGE_ID Function](#)
- [GET_PAGE_VIEW_ID Function](#)
- [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](#)

- [REMOVE_DEBUG_BY_APP Procedure](#)
- [REMOVE_DEBUG_BY_VIEW Procedure](#)
- [REMOVE_SESSION_MESSAGES Procedure](#)
- [TOCHAR Function](#)
- [TRACE Procedure](#)
- [WARN Procedure](#)

 **See Also:**

Accessing Debugging Mode in *Oracle APEX App Builder User's Guide*

23.1 Constants

The APEX_DEBUG package uses the following constants.

```
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;
    -- APEX engine: messages when procedures/functions are entered
c_log_level_engine_trace constant t_log_level := 9;
    -- APEX engine: other messages within procedures/functions
```

23.2 DISABLE Procedure

This procedure turns off debug messaging.

Syntax

```
APEX_DEBUG.DISABLE;
```

Parameters

None.

Example

This example turns off debug messaging.

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

 **See Also:**

[ENABLE Procedure](#)

23.3 DISABLE_DBMS_OUTPUT Procedure

This procedure stops writing all debug logs with `dbms_output`.

Syntax

```
DISABLE_DBMS_OUTPUT;
```

Parameters

None.

Example

See `ENABLE_DBMS_OUTPUT`.

 **See Also:**

- [ENABLE_DBMS_OUTPUT Procedure](#)
- [ENABLE Procedure](#)
- [DISABLE Procedure](#)

23.4 ENABLE Procedure

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

 **Note:**

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

Syntax

```
APEX_DEBUG.ENABLE (
    p_level      IN  t_log_level DEFAULT c_log_level_info )
```

Parameters

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

Example

This examples enables 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;
```

23.5 ENTER Procedure

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

Syntax

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

```
p_value10      IN VARCHAR2      DEFAULT NULL,
p_value_max_length IN PLS_INTEGER DEFAULT 1000 )
```

Parameters

Parameter	Description
p_routine_name	The name of the procedure or function.
p_namexx/p_valuexx	The procedure or function parameter name and value.
p_value_max_length	The p_valuexx values is truncated to this length. The tilde (~) character is appended to indicate that the original value exceeded this length.

Example

This example adds a debug message at the beginning of a procedure.

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

See Also:

- [MESSAGE Procedure](#)
- [ERROR Procedure](#)
- [WARN Procedure](#)
- [TRACE Procedure](#)
- [INFO Procedure](#)

23.6 ENABLE_DBMS_OUTPUT Procedure

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

Syntax

```
ENABLE_DBMS_OUTPUT (
    p_prefix      IN VARCHAR2      DEFAULT '# APEX|' )
```

Parameters

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

Example

This SQLcl 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_enable_dbms_output(p_prefix=>'Debug-');
    apex_debug.error('4');
    apex_debug.warn('5');
    apex_debug.info('6');
    apex_enable(p_level=>apex_debug.c_log_level_info);
    apex_debug.info('7');
    apex_enable_dbms_output;
    apex_debug.info('8');
    apex_disable_dbms_output;
    apex_debug.info('9');
end;
/
Output:
Debug-ERR|4
Debug-WRN|5
Debug-INF|7
# APEX|INF|8
```

See Also:

- [DISABLE_DBMS_OUTPUT Procedure](#)
- [ENABLE Procedure](#)
- [DISABLE Procedure](#)

23.7 ERROR Procedure

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

Syntax

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

Parameters

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

Example

This example logs a critical error in the debug log.

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

See Also:

- [MESSAGE Procedure](#)
- [ERROR Procedure](#)
- [WARN Procedure](#)
- [TRACE Procedure](#)
- [INFO Procedure](#)

23.8 GET_LAST_MESSAGE_ID Function

This function returns the identifier for the last debug message that was generated in this session. The value is null until the first debug message has been generated.

Syntax

```
APEX_DEBUG.GET_LAST_MESSAGE_ID (
    RETURN NUMBER );
```

Example

The following example prints the message identifiers before and after emitting debug output.

```
BEGIN
    sys.dbms_output.put_line('Page View ID='||apex_debug.get_last_message_id);
    apex_debug.message('Hello', p_force => true);
    sys.dbms_output.put_line('Page View ID='||apex_debug.get_last_message_id);
END;
```

23.9 GET_PAGE_VIEW_ID Function

This function returns the current page view identifier, which is a unique ID for each browser request or standalone database session. The value is null until the first debug message has been generated.

Syntax

```
APEX_DEBUG.GET_PAGE_VIEW_ID (
    RETURN NUMBER );
```

Example

The following example prints the page view identifiers before and after emitting debug output.

```
BEGIN
    sys.dbms_output.put_line('Page View ID='||apex_debug.get_page_view_id);
    apex_debug.message('Hello', p_force => true);
    sys.dbms_output.put_line('Page View ID='||apex_debug.get_page_view_id);
END;
```

23.10 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

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

Example

This example logs information in the debug log.

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

See Also:

- [MESSAGE Procedure](#)
- [ERROR Procedure](#)
- [WARN Procedure](#)
- [TRACE Procedure](#)
- [ENTER Procedure](#)

23.11 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 logs 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](#)
- [INFO Procedure](#)

23.12 LOG_LONG_MESSAGE Procedure

This procedure emits debug messages from PL/SQL components of Oracle APEX, or PL/SQL 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:**

As a best practice, Oracle recommends using shorter message APIs when possible (ERROR, WARN, and so on), and reserving LOG_LONG_MESSAGE for scenarios that require longer messages.

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

Parameter	Description
p_message	Log long message with maximum size of 32,767 bytes.

Parameter	Description
p_enabled	Set to TRUE to always log messages, irrespective of whether debugging is enabled. Set to FALSE to only log messages if debugging is enabled.
p_level	Identifies the level of the long log message. See Constants .

Example

This example enables debug message logging for level 1 and 2 messages and display a level 1 message that could contain anything up to 32,767 characters. Note the p_enabled parameter need not be specified, as debugging has been explicitly enabled and the default of FALSE for this parameter respects this enabling.

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

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

See Also:

- [ENTER Procedure](#)
- [ERROR Procedure](#)
- [INFO Procedure](#)
- [MESSAGE Procedure](#)
- [TRACE Procedure](#)
- [WARN Procedure](#)

23.13 LOG_MESSAGE Procedure (Deprecated)

This procedure logs a debug message.

 **Note:**

This API is deprecated and will be removed in a future release.

Instead of this procedure, use the following:

- [ERROR Procedure](#)
- [WARN Procedure](#)
- [MESSAGE Procedure](#)
- [INFO Procedure](#)
- [ENTER Procedure](#)
- [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

Parameter	Description
p_message	The debug message with a maximum length of 1,000 bytes.
p_enabled	Messages are logged when logging is enabled. TRUE enables logging.
p_level	Identifies the level of the log message where 1 is most important and 9 is least important. This is an integer value.

Example

This example enables debug message logging for level 1 and 2 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

23.14 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

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

Example

This example enables 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;
```

23.15 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

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

Example

This example adds 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](#)

23.16 REMOVE_DEBUG_BY_AGE Procedure

Deletes all data older than the specified number of days from the debug message log.

Syntax

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

Parameters

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

Example

This example removes 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;
```

23.17 REMOVE_DEBUG_BY_APP Procedure

Deletes all data belonging to a specified application from the debug message log.

Syntax

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

Parameters

Parameter	Description
p_application_id	The application ID of the application.

Example

This example removes all debug messages logged for the current application.

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

23.18 REMOVE_DEBUG_BY_VIEW Procedure

Deletes 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

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

Example

This example removes 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;
```

23.19 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

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

Example

This example removes all debug messages logged within the current session. Because no value is passed for the `p_session` parameter, the procedure defaults to the current session.

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

23.20 TOCHAR Function

This procedure converts a BOOLEAN to a VARCHAR2.

Syntax

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

Parameters

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

Example

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

```
DECLARE
    l_state boolean;
BEGIN
    ....
    apex_debug.info('Value of l_state is %s', apex_debug.tochar(l_state));
    ....
END;
```

23.21 TRACE Procedure

This procedure logs messages at level `c_log_level_app_trace`.

Syntax

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

Parameters

Parameter	Description
<code>p_message</code>	The debug message. Occurrences of <code>%</code> are replaced by <code>p0</code> to <code>p19</code> , as in <code>utl_lms.format_message</code> and C's <code>sprintf</code> . Occurrences of <code>%%</code> represent the special character <code>%</code> . Occurrences of <code>%<n></code> are replaced by <code>p<n></code> .
<code>p0</code> through <code>p9</code>	Substitution strings for <code>%</code> placeholders.
<code>p_max_length</code>	The <code>p<n></code> values are truncated to this length. The tilde (~) character is appended to indicate that the original value exceeded this length.

Example

This example logs 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](#)

23.22 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

Parameter	Description
<code>p_message</code>	The debug message. Occurrences of <code>%</code> are replaced by <code>p0</code> to <code>p19</code> , as in <code>utl_lms.format_message</code> and C's <code>sprintf</code> . Occurrences of <code>%%</code> represent the special character <code>%</code> . Occurrences of <code>%<n></code> are replaced by <code>p<n></code> .
<code>p0</code> through <code>p9</code>	Substitution strings for <code>%</code> placeholders.
<code>p_max_length</code>	The <code>p<n></code> values are truncated to this length. The tilde (~) character is appended to indicate that the original value exceeded this length.

Example

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

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

See Also:

- [MESSAGE Procedure](#)
- [ERROR Procedure](#)
- [ENTER Procedure](#)
- [TRACE Procedure](#)
- [INFO Procedure](#)

APEX_DG_DATA_GEN

This package contains the implementation for data generation in Oracle APEX.

 **Tip:**

The APIs in this package require an APEX session. See `APEX_SESSION` documentation for creating a session outside of the APEX App Builder context.

- [ADD_BLUEPRINT Procedure](#)
- [ADD_BLUEPRINT_FROM_FILE Procedure](#)
- [ADD_BLUEPRINT_FROM_TABLES Procedure](#)
- [ADD_COLUMN Procedure](#)
- [ADD_DATA_SOURCE Procedure](#)
- [ADD_TABLE Procedure](#)
- [EXPORT_BLUEPRINT Function](#)
- [GENERATE_DATA Procedure Signature 1](#)
- [GENERATE_DATA Procedure Signature 2](#)
- [GENERATE_DATA_INTO_COLLECTION Procedure](#)
- [GET_BLUEPRINT_ID Function](#)
- [GET_BP_TABLE_ID Function](#)
- [GET_EXAMPLE Function](#)
- [GET_WEIGHTED_INLINE_DATA Function](#)
- [IMPORT_BLUEPRINT Procedure](#)
- [PREVIEW_BLUEPRINT Procedure](#)
- [REMOVE_BLUEPRINT Procedure](#)
- [REMOVE_COLUMN Procedure](#)
- [REMOVE_DATA_SOURCE Procedure](#)
- [REMOVE_TABLE Procedure](#)
- [RESEQUENCE_BLUEPRINT Procedure](#)
- [STOP_DATA_GENERATION Procedure](#)
- [UPDATE_BLUEPRINT Procedure](#)
- [UPDATE_COLUMN Procedure](#)
- [UPDATE_DATA_SOURCE Procedure](#)
- [UPDATE_TABLE Procedure](#)

- [VALIDATE_BLUEPRINT Procedure](#)
- [VALIDATE_INSTANCE_SETTING Procedure](#)



See Also:

[APEX_SESSION](#)

24.1 ADD_BLUEPRINT Procedure

This procedure creates a blueprint which is a collection of tables with corresponding columns and data generation attributes.

Syntax

```
APEX_DG_DATA_GEN.ADD_BLUEPRINT (
    p_name          IN VARCHAR2,
    p_display_name IN VARCHAR2 DEFAULT NULL,
    p_description   IN VARCHAR2 DEFAULT NULL,
    p_lang          IN VARCHAR2 DEFAULT 'en',
    p_default_schema IN VARCHAR2 DEFAULT NULL,
    p_blueprint_id  OUT NUMBER )
```

Parameters

Parameter	Description
p_name	Identifier for the blueprint, combination of name and language is unique. Name is automatically upper cased and special characters removed.
p_display_name	Friendly display name.
p_description	Description of the blueprint.
p_lang	Blueprint language determines values from built-in data sources. If the built-in data source has 0 records in this language, en is used.
p_default_schema	The default schema name for the blueprint.
p_blueprint_id	ID of the added blueprint (OUT).

Example

```
DECLARE
    l_blueprint_id number;
BEGIN
    apex_dg_data_gen.add_blueprint(
        p_name      => 'Cars',
        p_display_name => 'My Cars Blueprint',
        p_description => 'A blueprint to generate car data',
        p_blueprint_id => l_blueprint_id);
END;
```

24.2 ADD_BLUEPRINT_FROM_FILE Procedure

This procedure imports a JSON blueprint from a workspace or application file. The file should be JSON, containing a correct blueprint definition.

Syntax

```
APEX_DG_DATA_GEN.ADD_BLUEPRINT_FROM_FILE (
    p_filename          IN VARCHAR2,                      -- name of workspace or
application file
    p_application_id   IN NUMBER  DEFAULT NULL,        -- Application ID of an
Application File, or null if a workspace file
    p_override_name    IN VARCHAR2 DEFAULT NULL,        -- Name of blueprint,
overrides the name provided in the file
    p_replace           IN BOOLEAN  DEFAULT FALSE,       -- return error if
blueprint exist and p_replace=FALSE
    p_blueprint_id     OUT NUMBER )
```

Parameters

Parameter	Description
p_filename	Name of the file (apex_application_files.filename).
p_application_id	ID of the application, or null for workspace files.
p_override_name	Name of blueprint, this will override the name provided in the file.
p_replace	Return error if blueprint exists and p_replace = FALSE. Will replace the blueprint (or p_override_name if provided).
p_blueprint_id	ID of the imported blueprint (OUT).

Example

```
DECLARE
    l_blueprint number;
BEGIN
    apex_dg_data_gen.add_blueprint_from_file
    (p_filename          => 'app/example.json',
     p_application_id   => 145,
     p_override_name    => 'My Application Blueprint',
     p_replace           => false,
     p_blueprint_id     => l_blueprint
    );
END;

DECLARE
    l_blueprint number;
BEGIN
    apex_dg_data_gen.add_blueprint_from_file
    (p_filename          => 'workspace/example.json',
     p_override_name    => 'My Workspace Blueprint',
     p_replace           => false,
     p_blueprint_id     => l_blueprint
    );
END;
```

```
    );
END;
```

24.3 ADD_BLUEPRINT_FROM_TABLES Procedure

This procedure creates a blueprint and adds the tables specified based on the data dictionary.

For all the table names specified by the user, the Data Generator retrieves each table from the current schema, plus its definition, column names, data types, and attributes as they come from the DB data dictionary.

Syntax

```
APEX_DG_DATA_GEN.ADD_BLUEPRINT_FROM_TABLES (
    p_name          IN VARCHAR2,
    p_tables        IN apex_t_varchar2,
    p_preserve_case IN VARCHAR2 DEFAULT 'N',
    p_exclude_columns IN apex_t_varchar2 DEFAULT c_empty_t_varchar2,
    p_description   IN VARCHAR2 DEFAULT NULL,
    p_lang          IN VARCHAR2 DEFAULT 'en',
    p_default_schema IN VARCHAR2 DEFAULT NULL,
    p_blueprint_id  OUT NUMBER );
```

Parameters

Parameter	Description
p_name	Name of blueprint, combination of name and language are unique. Name is automatically upper cased.
p_tables	List of tables and the number of records. The format is: apex_t_varchar2('<Table name>:[Rows]',...)
	For example: apex_t_varchar2('PRODUCTS:10','CUSTOMERS:50','SALES:1000') The ordering of tables should be: master tables before child tables (for FK relationships).
p_preserve_case	Defaults to N which forces table name to uppercase. If Y, preserves table case.
p_exclude_columns	String array (apex_t_varchar2) of column names to exclude from the auto column generation. The exclude columns parameter applies to all tables in the p_tables parameter.
p_description	Description of blueprint.
p_lang	Blueprint language determines values from built-in data sources. If the built-in data source has 0 records in this language, en is used.
p_default_schema	The default schema name for the blueprint.
p_blueprint_id	ID of the added blueprint (OUT).

Example

```

DECLARE
    l_blueprint_id number;
BEGIN
    apex_dg_data_gen.add_blueprint_from_tables(
        p_name          => 'Product Sales',
        p_tables        =>
apex_t_varchar2('PRODUCTS:10','CUSTOMERS:50','SALES:1000'),
        p_exclude_columns =>
apex_t_varchar2('CREATED_BY','CREATED_DATE'),
        p_description     => 'A blueprint to generate product sales
data',
        p_lang           => 'en',
        p_blueprint_id   => l_blueprint_id
    );
END;

```

24.4 ADD_COLUMN Procedure

This procedure adds a column to the blueprint table.

Syntax

```

APEX_DG_DATA_GEN.ADD_COLUMN (
    p_blueprint          IN VARCHAR2,
    p_sequence           IN PLS_INTEGER,
    p_table_name         IN VARCHAR2,
    p_column_name        IN VARCHAR2,
    p_preserve_case      IN VARCHAR2 DEFAULT 'N',
    p_display_name       IN VARCHAR2 DEFAULT NULL,
    p_max_length          IN NUMBER DEFAULT 4000,
    p_multi_value         IN VARCHAR2 DEFAULT 'N',
    p_mv_format           IN VARCHAR2 DEFAULT 'JSON',
    p_mv_unique            IN VARCHAR2 DEFAULT 'Y',
    p_mv_delimiter        IN VARCHAR2 DEFAULT ':',
    p_mv_min_entries      IN INTEGER DEFAULT 1,
    p_mv_max_entries      IN INTEGER DEFAULT 2,
    p_mv_partition_by     IN VARCHAR2 DEFAULT NULL,
    p_lang                IN VARCHAR2 DEFAULT 'en',
    p_data_source_type    IN VARCHAR2,
    p_data_source          IN VARCHAR2 DEFAULT NULL,
    p_ds_preserve_case    IN VARCHAR2 DEFAULT 'N',
    p_min_numeric_value    IN NUMBER DEFAULT 1,
    p_max_numeric_value    IN NUMBER DEFAULT 10,
    p_numeric_precision     IN NUMBER DEFAULT 0,
    p_min_date_value       IN DATE DEFAULT NULL,
    p_max_date_value       IN DATE DEFAULT NULL,
    p_format_mask          IN VARCHAR2 DEFAULT c_json_date_format,
    p_sequence_start_with  IN NUMBER DEFAULT 1,
    p_sequence_increment    IN NUMBER DEFAULT 1,
    p_formula              IN VARCHAR2 DEFAULT NULL,
    p_formula_lang          IN VARCHAR2 DEFAULT 'PLSQL',
    p_custom_attributes     IN VARCHAR2 DEFAULT NULL,
)

```

```
p_percent_blank          IN NUMBER      DEFAULT 0,
p_column_id              OUT NUMBER )
```

Parameters

Parameter	Description
p_blueprint	Identifier for the blueprint.
p_sequence	1 for first column, 2 for second, and so on.
p_table_name	Table name as known to the blueprint. Checks exact case first, then checks upper case.
p_column_name	Name of the column.
p_preserve_case	Defaults to N which forces column name to uppercase. If Y, preserves casing of parameter.
p_display_name	A friendly name for a given table.
p_max_length	When generating data (such as Latin text) substring to this.
p_multi_value	Y or N (currently available for BUILTIN table data and INLINE data). BUILTIN table data will be distinct. INLINE data will be distinct if all values appear once (red,1;blue,1;green,1). Otherwise, permits duplicates (red,3;blue,4;green,8). The number indicates the approximated frequency of each value on the generate data.
p_mv_format	DELIMITED (based upon p_mv_delimiter) or JSON (such as {"p_column_name" : ["sympton1", "sympton2"]}).
p_mv_unique	If Y, values do not repeat within the multi-value column. If N, indicates values may repeat.
p_mv_delimiter	Delimiter for a DELIMITED.
p_mv_min_entrie	Minimum values in a multi value list. s
p_mv_max_entrie	Maximum values in a multi value list. s
p_mv_partition_by	This value must match a column in the same built-in data source. For example, if p_data_source is "car.model", this value may be "make" because "car.make" is valid.
p_lang	Language code (for example en, de, es).
p_data_source_type	<ul style="list-style-type: none"> • BLUEPRINT • BUILTIN • DATA_SOURCE • FORMULA (requires p_data_source to be NULL) • INLINE • SEQUENCE

Parameter	Description
p_data_source	<p>Can be set to one of the following options:</p> <ul style="list-style-type: none"> • DATA_SOURCE: DATA_SOURCE_NAME.COLUMN_NAME (column name's case follows p_ds_preserve_case and defaults to upper case). • BUILTIN: see built-in list, must match a built-in exactly. • BLUEPRINT: references table data already generated (table must have lower sequence). For example, Dept.Deptno where add_table with p_table_name = Dept and add_column with Deptno exist.
	<p> Note:</p> <p>This is case-sensitive. Tables created with p_preserve_case = N are automatically uppercased. May require DEPT.DEPTNO instead of dept.deptno).</p>
	<ul style="list-style-type: none"> • INLINE: PART_TIME,3;FULL_TIME,7
	<p> Note:</p> <p>Inline format is VALUE,FREQUENCY, separated by a semi-colon. The frequency of the value is an approximation and Oracle best practice is to use the smallest numeric values that provide the desired distribution.</p>
	<ul style="list-style-type: none"> • SEQUENCE: uses p_sequence_parameters. • FORMULA: p_data_source must be NULL. Uses p_formula as a PL/SQL formula and {column_name} as substitutions from this table. For example, p_formula => {first_name} '.' {last_name} '.insum.ca'
p_ds_preserve_case	If p_data_source_type in ('DATA_SOURCE', 'BLUEPRINT') and p_ds_preserve_case = N, then the data source is upper cased to match an upper case table_name.column_name
p_min_numeric_value	A positive integer number used as the minimum value (inclusive) to be used in BUILTIN data sources that return NUMBER values.
p_max_numeric_value	A positive integer number used as the maximum value (inclusive) to be used in BUILTIN data sources that return NUMBER values.
p_numeric_precision	0 = no decimal values -1 = round to ten positive integer = number of decimal places
p_min_date_value	A DATE used as the minimum value (inclusive) to be used in BUILTIN data sources that return DATE type values.
p_max_date_value	A DATE used as the maximum value (inclusive) to be used in BUILTIN data sources that return DATE type values.
p_format_mask	Format mask when datatype is a date.
p_sequence_start_with	Only used when p_data_source_type = SEQUENCE.
p_sequence_increment	Only used when p_data_source_type = SEQUENCE.

Parameter	Description
p_formula	Enables referencing columns in this row, PL/SQL expressions that can reference columns defined in this blueprint row. For example: {FIRST_NAME} '.' {LAST_NAME} '.insum.ca' Substitutions are case sensitive and must match {column_name} exactly. If p_preserve_case was set to N, {COLUMN_NAME} must be upper case. Can be used on any DATA_SOURCE_TYPE. Formulas are applied last, after p_percent_blank. If p_percent_blank = 100 but FORMULAR is sysdate, the column value will be sysdate.
p_formula_lang	Formulas can be used as a combination of PL/SQL functions performed on this or other columns using {column_name} notation. String/Char, Date/Time, Numeric/Math functions are supported.
p_custom_attributes	For future expansion.
p_percent_blank	0 to 100. This is applied prior to all formulas. If this column is referenced in a formula, the formula contains a blank when appropriate.
<p> Note:</p> <p>A formula on this column may cause the column to not be blank.</p>	
p_column_id	ID of the added column (OUT).

Example

```

DECLARE
    l_column_id number;
BEGIN
    apex_dg_data_gen.add_column(
        p_blueprint          => 'Cars',
        p_sequence           => 1,
        p_table_name         => 'MY_CARS',
        p_column_name        => 'make',
        p_data_source_type   => 'BUILTIN',
        p_data_source        => 'car.make',
        p_column_id          => l_column_id);
END;

```

24.5 ADD_DATA_SOURCE Procedure

This procedure creates a data source which identifies a table or query from which you can source data values.

Syntax

```
APEX_DG_DATA_GEN.ADD_DATA_SOURCE (
    p_blueprint          IN VARCHAR2,
    p_name               IN VARCHAR2,
    p_data_source_type   IN VARCHAR2,
    p_table              IN VARCHAR2 DEFAULT NULL,
    p_preserve_case      IN VARCHAR2 DEFAULT 'N',
    p_sql_query          IN VARCHAR2 DEFAULT NULL,
    p_where_clause       IN VARCHAR2 DEFAULT NULL,
    p_inline_data        IN CLOB    DEFAULT NULL,
    p_order_by_column    IN VARCHAR2 DEFAULT NULL,
    p_data_source_id     OUT NUMBER )
```

Parameters

Parameter	Description
p_blueprint	Identifies the blueprint.
p_name	Name of a data source. Name is upper cased and must be 26 characters or less.
p_data_source_type	TABLE or SQL_QUERY.
p_table	For source type = TABLE. Typically this will match p_name.
p_preserve_case	Defaults to N which forces p_table_name to uppercase, if Y preserves casing of p_table.
p_sql_query	For p_data_source_type = SQL_QUERY.
p_where_clause	For p_data_source_type = TABLE, this adds the where clause. Do not include "where" keyword (for example, deptno <= 20).
p_inline_data	For p_data_source_type = JSON_DATA.
p_order_by_column	Not used.
p_data_source	The ID of the added data source (OUT).

Example

```
DECLARE
    l_data_source_id number;
BEGIN
    apex_dg_data_gen.add_data_source(
        p_blueprint          => 'Cars',
        p_name               => 'apex_dg_builtin_cars',
        p_data_source_type   => 'TABLE',
        p_table              => 'apex_dg_builtin_cars',
        p_data_source_id     => l_data_source_id );
END;
```

24.6 ADD_TABLE Procedure

This procedure adds a destination table for the generated data.

Syntax

```
APEX_DG_DATA_GEN.ADD_TABLE (
    p_blueprint          IN VARCHAR2,
    p_sequence           IN PLS_INTEGER,
    p_table_name         IN VARCHAR2,
    p_preserve_case     IN VARCHAR2          DEFAULT 'N',
    p_display_name       IN VARCHAR2          DEFAULT NULL,
    p_singular_name      IN VARCHAR2          DEFAULT NULL,
    p_plural_name        IN VARCHAR2          DEFAULT NULL,
    p_rows               IN NUMBER            DEFAULT 0,
    p_max_rows           IN NUMBER            DEFAULT NULL,
    p_use_existing_table IN VARCHAR2          DEFAULT 'N',
    p_exclude_columns    IN apex_t_varchar2  DEFAULT
c_empty_t_varchar2,
    p_table_id           OUT NUMBER )
```

Parameters

Parameter	Description
p_blueprint	Identifier for the blueprint.
p_sequence	1 for first table, 2 for second, and so forth.
p_table_name	Name of table that can exist or not exist.
p_preserve_case	Defaults to N, which forces table name to uppercase. If Y, preserves casing of parameter.
p_display_name	Friendly display name.
p_singular_name	Singluar friendly name.
p_plural_name	Plural friendly name.
p_rows	Number of rows to generate for this table.
p_max_rows	If null, then p_rows determines the number of rows, otherwise random rows between p_rows and p_max_rows are used when generating output.
p_use_existing_table	If Y, uses the data dictionary to auto generate columns. The automatic blueprint column creation supports the following data source mapping rules: <ul style="list-style-type: none"> Foreign key data generation (populates the column with keys from the master table). Inline data generation based on CHECK constraints (simple IN constructs are supported). Mapping based on existing built-in tables (based on the table and column name). Mapping based on the column name, data type, and length. If the column is nullable, 5% of the values will be NULL.
p_exclude_columns	String array (apex_t_varchar2) of column names to exclude from the automatic column generation.
p_table_id	ID of the added table (OUT).

Example

```

DECLARE
    l_table_id number;
BEGIN
    apex_dg_data_gen.add_table(
        p_blueprint      => 'Cars',
        p_sequence       => 1,
        p_table_name     => 'my_cars',
        p_rows           => '50',
        p_table_id       => l_table_id);

    apex_dg_data_gen.add_table(
        p_blueprint      => 'Cars',
        p_sequence       => 1,
        p_table_name     => 'my_cars',
        p_rows           => '50',
        p_use_existing_table => 'Y',
        p_table_id       => l_table_id
    );

    apex_dg_data_gen.add_table(
        p_blueprint      => 'Cars',
        p_sequence       => 1,
        p_table_name     => 'my_cars',
        p_rows           => '50',
        p_use_existing_table => 'Y',
        p_exclude_columns =>
    apex_t_varchar2('CREATED_BY','CREATED_DATE'),
        p_table_id       => l_table_id
    );
END;

```

24.7 EXPORT_BLUEPRINT Function

This function exports a blueprint in JSON format.

Syntax

```

APEX_DG_DATA_GEN.EXPORT_BLUEPRINT (
    p_name          IN VARCHAR2,
    p_pretty        IN VARCHAR2 DEFAULT 'Y' )
RETURN CLOB;

```

Parameters

Parameter	Description
p_name	Name of blueprint to export.
p_pretty	Y to return pretty results, all other values do not.

Returns

Returns the blueprint as a JSON document in a CLOB.

Example

```
DECLARE
    l_json clob;
BEGIN
    l_json := apex_dg_data_gen.export_blueprint(
        p_name => 'Cars');
END;
```

24.8 GENERATE_DATA Procedure Signature 1

This procedure creates rows of data based on the blueprint tables and their columns customizations.

This procedure inserts data into tables in the schema when the `p_format` is set to `INSERT INTO` or `FAST INSERT INTO`. The outputs do not contain data (all are set to NULL).

This procedure also generates data in a file. For that file, the three outputs contain the following data:

- `p_output` (BLOB) with the data output. Contents can be inside a JSON, CSV, ZIP, or SQL file.
- `p_file_ext` and `p_mime_type` (VARCHAR2) indicates the file extension and its MIME type.

These three output parameters send the file to the user's browser so it can be handled client-side.

In both scenarios, `p_errors` may have a NULL value or a CLOB with a JSON output that contains any errors.

Syntax

```
APEX_DG_DATA_GEN.GENERATE_DATA (
    p_blueprint          IN      VARCHAR2,
    p_format             IN      VARCHAR2,
    p_blueprint_table   IN      VARCHAR2 DEFAULT NULL,
    p_row_scaling        IN      NUMBER DEFAULT 100,
    p_stop_after_errors IN      NUMBER DEFAULT c_max_error_count,
    p_output              OUT NOCOPY BLOB,
    p_file_ext            OUT NOCOPY VARCHAR2,
    p_mime_type           OUT NOCOPY VARCHAR2,
    p_errors              OUT NOCOPY CLOB )
```

Parameters

Parameter	Description
<code>p_blueprint</code>	Name of the blueprint.

Parameter	Description
p_format	Can be set to one of the following options: SQL INSERT outputs a SQL script. CSV outputs a single CSV for one table or a ZIP of CSVs for multiple tables. JSON outputs JSON file. INSERT INTO generates the SQL INSERT script and runs the insert statements in the current schema. FAST INSERT INTO generates the data and does a single INSERT ... into [table] SELECT ... from [temporary table].
p_blueprint_table	Null for all tables. If not null, generates data only for designated table. If not null, must be table name of a table within the blueprint. This value is case sensitive.
p_row_scaling	Scales the number of rows defined into the blueprint by this percentage value.
p_stop_after_errors	How many errors can happen before the process is stopped. This is only applicable for INSERT INTO.
p_output	The blob to hold the output. Null for INSERT INTO and FAST INSERT INTO.
p_file_ext	The file extension of the output. Null for INSERT INTO and FAST INSERT INTO.
p_mime_type	The MIME type of the output. Null for INSERT INTO and FAST INSERT INTO.
p_errors	JSON output of any errors. NULL upon success.

Example

```

DECLARE
    l_output      blob;
    l_file_ext   varchar2(255);
    l_mime_type  varchar2(255);
    l_errors      clob;
BEGIN
    apex_dg_output.generate_data
        (p_blueprint          => 'Cars',
         p_blueprint_table    => 'my_cars',
         p_stop_after_errors => 100,
         p_output              => l_output
         p_file_ext            => l_file_ext,
         p_mime_type           => l_mime_type,
         p_errors               => l_errors
        );
END;

```

24.9 GENERATE_DATA Procedure Signature 2

This procedure creates rows of data based on the blueprint tables and their columns customizations.

This procedure inserts data into user-specified tables in the schema when the `p_format` is set to `INSERT INTO` or `FAST INSERT INTO`. The outputs do not contain data (all are set to NULL).

This procedure also generates data in a file. For that file, the three outputs contain the following data:

- `p_output` (BLOB) with the data output. Contents can be inside a JSON, CSV, ZIP, or SQL file.
- `p_file_ext` and `p_mime_type` (VARCHAR2) indicates the actual file extension and its MIME type.

These three output parameters send the file to the user's browser so it can be handled client-side.

In both scenarios, `p_errors` may have a NULL value or a CLOB with a JSON output that contains any errors.

Syntax

```
APEX_DG_DATA_GEN.GENERATE_DATA (
    p_blueprint          IN      VARCHAR2,
    p_format             IN      VARCHAR2,
    p_blueprint_table   IN      VARCHAR2 DEFAULT NULL,
    p_row_scaling        IN      NUMBER DEFAULT 100,
    p_stop_after_errors IN      NUMBER DEFAULT c_max_error_count,
    p_output              OUT NOCOPY CLOB,
    p_file_ext            OUT NOCOPY VARCHAR2,
    p_mime_type           OUT NOCOPY VARCHAR2,
    p_errors              OUT NOCOPY CLOB )
```

Parameters

Parameter	Description
<code>p_blueprint</code>	Name of the blueprint.
<code>p_format</code>	Can be set to one of the following options: <code>SQL</code> <code>INSERT</code> outputs a SQL script. <code>CSV</code> outputs a single CSV for one table or a ZIP of CSVs for multiple tables. <code>JSON</code> outputs JSON file. <code>INSERT INTO</code> generates the SQL <code>INSERT</code> script and runs the <code>insert</code> statements in the current schema. <code>FAST INSERT INTO</code> generates the data and does a single <code>INSERT ... into [table] SELECT ... from [temporary table]</code> .
<code>p_blueprint_table</code>	Null for all tables. If not null, will generate data only for designated table. If not null, must be table name of a table within the blueprint. Note: this value is case sensitive.
<code>p_row_scaling</code>	Will scale the number of rows defined into the blueprint by this percentage value.
<code>p_stop_after_errors</code>	How many errors can happen before the process is stopped. This is only applicable for <code>INSERT INTO</code>
<code>p_output</code>	The clob to hold the output. Null for <code>INSERT INTO</code> and <code>FAST INSERT INTO</code> .

Parameter	Description
p_file_ext	The file extension of the output. Null for INSERT INTO and FAST INSERT INTO.
p_mime_type	The MIME type of the output. Null for INSERT INTO and FAST INSERT INTO.
p_errors	JSON output of any errors. NULL upon success.

Example

```

DECLARE
    l_output      clob;
    l_file_ext   varchar2(255);
    l_mime_type  varchar2(255);
    l_errors      clob;
BEGIN
    apex_dg_output.generate_data
        (p_blueprint      => 'Cars',
         p_blueprint_table => 'my_cars',
         p_stop_after_errors => 100,
         p_output          => l_output
         p_file_ext       => l_file_ext,
         p_mime_type      => l_mime_type,
         p_errors          => l_errors
        );
END;

```

24.10 GENERATE_DATA_INTO_COLLECTION Procedure

This procedure generates the data of the specified blueprint and stores the results in an APEX collection named APEX\$DG\$[BLUEPRINT_NAME].

Syntax

```

APEX_DG_DATA_GEN.GENERATE_DATA_INTO_COLLECTION (
    p_blueprint      IN      VARCHAR2,
    p_format         IN      VARCHAR2,
    p_blueprint_table IN      VARCHAR2 DEFAULT NULL,
    p_row_scaling    IN      NUMBER DEFAULT 100,
    p_stop_after_errors IN      NUMBER DEFAULT c_max_error_count,
    p_errors         OUT NOCOPY CLOB )

```

Parameters

Parameter	Description
p_blueprint	Name of the blueprint.

Parameter	Description
p_format	SQL INSERT outputs a sql script. CSV outputs a single CSV for one table or a ZIP of CSVs for multiple tables. JSON outputs JSON file. INSERT INTO generates the SQL INSERT script and runs the insert statements in the current schema. FAST INSERT INTO generates the data and does a single INSERT ... into [table] SELECT ... from [temporary table]
p_blueprint_table	This value is case sensitive. Null for all tables. If not null, generates data only for designated table. If not null, must be table name of a table within the blueprint.
p_row_scaling	Scales the number of rows defined into the blueprint by this percentage value.
p_stop_after_errors	Defines the number of errors that can happen before the process is stopped. Only applies to INSERT INTO.
p_errors	JSON output of any errors. NULL upon success.

Output is stored in the collection. Additionally, a new row within the same collection contains the error message if there is none.

Output	Description
CLOB001	The blob to hold the output. Null for INSERT INTO and FAST INSERT INTO.
BLOB001	The blob to hold the output. Null for INSERT INTO and FAST INSERT INTO.
C006	The file extension of the output. Null for INSERT INTO and FAST INSERT INTO.
C007	The mime type of the output. Null for INSERT INTO and FAST INSERT INTO.
C001	'ERROR'
CLOB001	JSON output of any errors. NULL upon success.

Example

```
DECLARE
    l_errors      clob;
BEGIN
    apex_dg_output.generate_data_into_collection
        (p_blueprint          => 'Cars',
         p_blueprint_table   => 'my_cars',
         p_stop_after_errors => 100,
         p_errors            => l_errors
        );
END;
```

24.11 GET_BLUEPRINT_ID Function

This function returns the blueprint ID from the name.

Syntax

```
APEX_DG_DATA_GEN.GET_BLUEPRINT_ID (
    p_name    IN VARCHAR2 )
    RETURN NUMBER;
```

Parameters

Parameter	Description
p_name	The blueprint identifier.

Returns

ID of the blueprint.

Example

The following example demonstrates

```
DECLARE
    l_blueprint_id apex_dg_blueprints.id%TYPE;
BEGIN
    l_blueprint_id := apex_dg_data_gen.get_blueprint_id(p_name => 'MY
BLUEPRINT');
END;
```

24.12 GET_BP_TABLE_ID Function

This function returns the `table_id` for a given blueprint ID and table name (case-sensitive). If not found, it searches with UPPERCASE `p_table_name` automatically.

Syntax

```
APEX_DG_DATA_GEN.GET_BP_TABLE_ID (
    p_bp_id      IN NUMBER,
    p_table_name IN VARCHAR2 )
    RETURN NUMBER;
```

Parameters

Parameter	Description
p_bp_id	The blueprint ID.
p_table_name	The name of the table.

Returns

The table ID.

Example

```

DECLARE
    v_table_id number;
BEGIN
    l_table_id := apex_dg_data_gen.get_bp_table_id
        (p_bp_id      => 12345,
         p_table_name => 'DEPT'
        );
END;

```

24.13 GET_EXAMPLE Function

This function generates example data for the friendly name of built-in data. The function returns a (user-specified) number of examples, showing the data to expect when using this friendly name.

Syntax

```

APEX_DG_DATA_GEN.GET_EXAMPLE (
    p_friendly_name    IN VARCHAR2,
    p_lang             IN VARCHAR2 DEFAULT 'en',
    p_rows             IN NUMBER DEFAULT 5 )
RETURN wwv_flow_t_varchar2;

```

Parameters

Parameter	Description
p_friendly_name	The friendly name.
p_lang	(Optional) The language.
p_rows	Number of rows (examples) to return.

Example

The following example returns five rows from the domain of values for the built-in with the friendly name `animal.family`.

```

select *
from apex_dg_data_gen.get_example( p_friendly_name => 'animal.family');

```

24.14 GET_WEIGHTED_INLINE_DATA Function

This function returns a list of generated inline rows from a semi colon (;) delimited list of values. For each value add a comma to define weight (such as F,45;M,30).

Syntax

```

APEX_DG_DATA_GEN.GET_WEIGHTED_INLINE_DATA (
    p_data  IN VARCHAR2 )
RETURN wwv_flow_t_varchar2

```

Parameters

Parameter	Description
p_data	The list of values.

Example

The following example returns two rows: F and M.

```
select *
from apex_dg_data_gen.get_weighted_inline_data( p_data => 'F;M');
```

24.15 IMPORT_BLUEPRINT Procedure

This procedure imports a JSON blueprint.

Syntax

```
APEX_DG_DATA_GEN.IMPORT_BLUEPRINT (
    p_clob          IN CLOB,
    p_override_name IN VARCHAR2 DEFAULT NULL,
    p_replace       IN BOOLEAN DEFAULT FALSE,
    p_blueprint_id OUT NUMBER )
```

Parameters

Parameter	Description
p_clob	Blueprint in JSON format.
p_override_name	Name of blueprint. This overrides the name provided in p_clob.
p_replace	Return error if blueprint exists and p_replace is FALSE. Replaces the blueprint (or p_override_name if provided).
p_blueprint_id	ID of the imported blueprint (OUT).

Example

```
DECLARE
    l_json clob;
    l_blueprint_id number;
BEGIN
    l_json := apex_dg_data_gen.export_blueprint(
        p_name => 'Cars');

    apex_dg_data_gen.import_blueprint(
        p_clob => l_json,
        p_replace => TRUE,
        p_blueprint_id => l_blueprint_id);
END;
```

24.16 PREVIEW_BLUEPRINT Procedure

This procedure creates preview data for a blueprint and stores this in APEX collections. This procedure can only be used with an active APEX session.

Syntax

```
APEX_DG_DATA_GEN.PREVIEW_BLUEPRINT (
    parameter_1 IN NUMBER,
    parameter_2 IN VARCHAR2,
    parameter_3 IN NUMBER )
```

Parameters

Parameter	Description
p_blueprint	Name of the blueprint.
p_table_name	If null, all tables. If not null, the specified table.
p_number_of_rows	Number of rows to generate (maximum of 50).
p_data_collection	Name of the APEX collection for data.
p_header_collection	Name of the APEX collection for headers.

Example

```
BEGIN
    apex_dg_output.preview_blueprint
        (p_blueprint      => 'Cars',
         p_table_name    => 'my_cars',
         p_data_collection => 'CARS_DATA',
         p_header_collection => 'CARS_HEADERS'
        );
END;
```

24.17 REMOVE_BLUEPRINT Procedure

This procedure removes metadata associated with a blueprint.

Syntax

```
APEX_DG_DATA_GEN.REMOVE_BLUEPRINT (
    p_name           IN VARCHAR2 )
```

Parameters

Parameter	Description
p_name	Name of blueprint to be removed.

Example

```
BEGIN
    apex_dg_data_gen.remove_blueprint(
        p_name          => 'Cars');
END;
```

24.18 REMOVE_COLUMN Procedure

This procedure removes a column from the blueprint table.

Syntax

```
APEX_DG_DATA_GEN.REMOVE_COLUMN (
    p_blueprint      IN VARCHAR2,
    p_table_name     IN VARCHAR2,
    p_column_name    IN VARCHAR2 )
```

Parameters

Parameter	Description
p_blueprint	Identifier for the blueprint.
p_table_name	Name of table within blueprint.
p_column_name	Name of column within table.

Example

```
BEGIN
    apex_dg_data_gen.remove_column(
        p_blueprint      => 'Cars',
        p_table_name     => 'MY_CARS',
        p_column_name    => 'MAKE');
END;
```

24.19 REMOVE_DATA_SOURCE Procedure

This procedure removes metadata associated with the data source for the given blueprint.

Syntax

```
APEX_DG_DATA_GEN.REMOVE_DATA_SOURCE (
    p_blueprint      IN VARCHAR2,
    p_name           IN VARCHAR2 )
```

Parameters

Parameter	Description
p_blueprint	Identifies the blueprint.

Parameter	Description
p_name	Data source to be removed from blueprint.

Example

```
BEGIN
    apex_dg_data_gen.remove_data_source(
        p_blueprint      => 'Cars',
        p_name           => 'apex_dg_builtin_cars');
END;
```

24.20 REMOVE_TABLE Procedure

This procedure removes a table for the specified blueprint.

Syntax

```
APEX_DG_DATA_GEN.REMOVE_TABLE (
    p_blueprint      IN VARCHAR2,
    p_table_name     IN VARCHAR2 )
```

Parameters

Parameter	Description
p_blueprint	Identifier for the blueprint.
p_table_name	Table name to be removed from blueprint.

Example

```
BEGIN
    apex_dg_data_gen.remove_table(
        p_blueprint      => 'Cars',
        p_table_name     => 'MY_CARS');
END;
```

24.21 RESEQUENCE_BLUEPRINT Procedure

This procedure resequences all tables and columns within tables with gaps of p_offset, retaining their current order.

Syntax

```
APEX_DG_DATA_GEN.RESEQUENCE_BLUEPRINT (
    p_blueprint IN VARCHAR2,
    p_offset     IN NUMBER    DEFAULT c_default_seq_offset )
```

Parameters

Parameter	Description
p_blueprint	Identifier for the blueprint.
p_offset	The offset between gaps, such as 10, 100, or 1000.

Example

```
BEGIN
    apex_dg_data_gen.resequence_blueprint(
        p_blueprint      => 'Cars',
        p_offset         => 100);
END;
```

24.22 STOP_DATA_GENERATION Procedure

This procedure stops the current data generation process. This only works within an Oracle APEX session.

This procedure relies on an APEX Collection which tracks progress and reacts to stop instructions. The collection is named: APEX\$DG\$[BLUEPRINT_NAME] and contains the following attributes:

```
d001 => current_timestamp of the process step
c001 => Blueprint name
c002 => Requested output format
c003 => Table name being generated
c004 => Name of the process step,
c005 => Description of the process step
n001 => Numeric identifier of the process step
```

Syntax

```
APEX_DG_DATA_GEN.STOP_DATA_GENERATION (
    p_blueprint      IN VARCHAR2 )
```

Parameters

Parameter	Description
p_blueprint	Name of the blueprint.

Example

```
BEGIN
    apex_dg_output.stop_data_generation
        (p_blueprint      => 'CARS',
        );
END;
```

24.23 UPDATE_BLUEPRINT Procedure

This procedure updates the attributes of an existing blueprint.

Syntax

```
APEX_DG_DATA_GEN.UPDATE_BLUEPRINT (
    p_name          IN VARCHAR2,
    p_new_name      IN VARCHAR2 DEFAULT NULL,
    p_display_name IN VARCHAR2 DEFAULT NULL,
    p_description   IN VARCHAR2 DEFAULT NULL,
    p_lang          IN VARCHAR2 DEFAULT 'en',
    p_default_schema IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_name	Name of blueprint to update.
p_new_name	The new name (rename). The name is upper cased and special characters removed.
p_display_name	Friendly display name.
p_description	Description of the blueprint.
p_lang	Blueprint language determines values from built-in data sources. If the built-in data source has 0 records in this language, en is used.

Example

```
BEGIN
    apex_dg_data_gen.update_blueprint(
        p_name          => 'Cars',
        p_display_name  => 'My Cars',
        p_description   => 'An updated blueprint to generate car
data');
END;
```

24.24 UPDATE_COLUMN Procedure

This procedure updates an existing column in a blueprint table.

Syntax

```
APEX_DG_DATA_GEN.UPDATE_COLUMN (
    p_blueprint      IN VARCHAR2,
    p_table_name     IN VARCHAR2,
    p_column_name    IN VARCHAR2,
    p_new_column_name IN VARCHAR2      DEFAULT NULL,
    p_sequence       IN PLS_INTEGER,
    p_preserve_case IN VARCHAR2      DEFAULT 'N',
    p_display_name   IN VARCHAR2      DEFAULT NULL,
```

```

    p_max_length          IN NUMBER      DEFAULT 4000,
    p_multi_value         IN VARCHAR2   DEFAULT 'N',
    p_mv_format            IN VARCHAR2   DEFAULT 'JSON',
    p_mv_unique             IN VARCHAR2   DEFAULT 'Y',
    p_mv_delimiter          IN VARCHAR2   DEFAULT ':',
    p_mv_min_entries        IN INTEGER     DEFAULT 1,
    p_mv_max_entries        IN INTEGER     DEFAULT 2,
    p_mv_partition_by       IN VARCHAR2   DEFAULT NULL,
    p_lang                  IN VARCHAR2   DEFAULT 'en',
    p_data_source_type      IN VARCHAR2,
    p_data_source           IN VARCHAR2   DEFAULT NULL,
    p_ds_preserve_case      IN VARCHAR2   DEFAULT 'N',
    p_min_numeric_value     IN NUMBER     DEFAULT 1,
    p_max_numeric_value     IN NUMBER     DEFAULT 10,
    p_numeric_precision      IN NUMBER     DEFAULT 0,
    p_min_date_value        IN DATE       DEFAULT NULL,
    p_max_date_value        IN DATE       DEFAULT NULL,
    p_format_mask            IN VARCHAR2   DEFAULT c_json_date_format,
    p_sequence_start_with    IN NUMBER     DEFAULT 1,
    p_sequence_increment     IN NUMBER     DEFAULT 1,
    p_formula                IN VARCHAR2   DEFAULT NULL,
    p_formula_lang            IN VARCHAR2   DEFAULT 'PLSQL',
    p_custom_attributes       IN VARCHAR2   DEFAULT NULL,
    p_percent_blank           IN NUMBER    DEFAULT 0 )
  
```

Parameters

Parameter	Description
p_blueprint	Identifier for the blueprint.
p_table_name	Table name as known to the blueprint. Checks exact case first, then checks upper case.
p_column_name	Name of the column.
p_new_column_name	New name of column (rename).
p_sequence	1 for first column, 2 for second, and so on.
p_preserve_case	Defaults to N which forces column name to uppercase. If Y, preserves casing of parameter.
p_display_name	A friendly name for a given table.
p_max_length	When generating data (such as Latin text) substring to this.
p_multi_value	Y or N (currently available for BUILTIN table data and INLINE data). BUILTIN table data will be distinct. INLINE data will be distinct if all values appear once (red,1;blue,1;green,1). Otherwise, permits duplicates (red,3;blue,4;green,8). The number indicates the approximated frequency of each value on the generate data.
p_mv_format	DELIMITED (based upon p_mv_delimiter) or JSON (such as {"p_column_name" : ["sympton1","sympton2"]}).
p_mv_unique	If Y, values do not repeat within the multi-value column. If N, indicates values may repeat.
p_mv_delimiter	Delimiter for a DELIMITED.
p_mv_min_entries	Minimum values in a multi value list.
p_mv_max_entries	Maximum values in a multi value list.

Parameter	Description	
p_mv_partition_by	This value must match a column in the same built-in data source. For example, if p_data_source is "car.model", this value may be "make" because "car.make" is valid.	
p_lang	Language code (for example en, de, es).	
p_data_source_type	<ul style="list-style-type: none"> • BLUEPRINT • BUILTIN • DATA_SOURCE • FORMULA (requires p_data_source to be null) • INLINE • SEQUENCE 	
p_data_source	<p>When p_data_source_type = DATA_SOURCE then DATA_SOURCE_NAME.COLUMN_NAME (column name's case follows p_ds_preserve_case and defaults to upper case).</p> <p>Can be set to one of the following options:</p> <ul style="list-style-type: none"> • BUILTIN: see built-in list, must match a built-in exactly. • BLUEPRINT: references table data already generated (table must have lower sequence). For example, Dept.Deptno where add_table with p_table_name = Dept and add_column with Deptno exist. 	
 Note:	<p>This is case-sensitive. Tables created with p_preserve_case = N are automatically uppercased. May require DEPT.DEPTNO instead of dept.deptno).</p>	
• INLINE: PART_TIME, 3;FULL_TIME, 7	 Note:	<p>Inline format is VALUE,FREQUENCY, separated by a semi-colon. The frequency of the value is an approximation and Oracle best practice is to use the smallest numeric values that provide the desired distribution.</p>
• SEQUENCE: uses p_sequence_parameters.	 Note:	<p>• FORMULA: p_data_source must be null. Uses p_formula as a PL/SQL formula and {column_name} as substitutions from this table. For example, p_formula => {first_name} '.' {last_name} '.insum.ca'</p>
p_ds_preserve_case	<p>If p_data_source_type in ('DATA_SOURCE', 'BLUEPRINT') and p_ds_preserve_case = N, then the data source is upper cased to match an upper case table_name.column_name</p>	
p_min_numeric_value	<p>A positive integer number used as the minimum value (inclusive) to be used in BUILTIN data sources that return NUMBER values.</p>	
p_max_numeric_value	<p>A positive integer number used as the maximum value (inclusive) to be used in BUILTIN data sources that return NUMBER values.</p>	

Parameter	Description
p_numeric_precision	0 = no decimal values -1 = round to ten positive integer = number of decimal places
p_min_date_value	A DATE used as the minimum value (inclusive) to be used in BUILTIN data sources that return DATE type values.
p_max_date_value	A DATE used as the maximum value (inclusive) to be used in BUILTIN data sources that return DATE type values.
p_format_mask	Format mask when datatype is a date.
p_sequence_start_with	Only used when p_data_source_type = SEQUENCE.
p_sequence_increment	Only used when p_data_source_type = SEQUENCE.
p_formula	Enables referencing columns in this row, PL/SQL expressions that can reference columns defined in this blueprint row. For example: {FIRST_NAME} '.' {LAST_NAME} '.insum.ca'
	Substitutions are case sensitive and must match {column_name} exactly. If p_preserve_case was set to N, {COLUMN_NAME} must be upper case. Can be used on any DATA_SOURCE_TYPE. Formulas are applied last, after p_percent_blank. If p_percent_blank = 100 but FORMULAR is sysdate, the column value will be sysdate.
p_formula_lang	Formulas can be used as a combination of PL/SQL functions performed on this or other columns using {column_name} notation. String/Char, Date/Time, Numeric/Math functions are supported.
p_custom_attributes	For future expansion.
p_percent_blank	0 to 100. This is applied prior to all formulas. If this column is referenced in a formula, the formula contains a blank when appropriate.

Note:

A formula on this column may cause the column to **not** be blank.

Example

```
BEGIN
    apex_dg_data_gen.update_column(
        p_blueprint            => 'Cars',
        p_sequence              => 1,
        p_table_name            => 'MY_CARS',
        p_column_name           => 'make',
        p_data_source_type      => 'BUILTIN',
        p_data_source            => 'car.make');
END;
```

24.25 UPDATE_DATA_SOURCE Procedure

This procedure updates an existing data source which identifies a table or query from which you can source data values.

Syntax

```
APEX_DG_DATA_GEN.UPDATE_DATA_SOURCE (
    p_blueprint          IN VARCHAR2,
    p_name               IN VARCHAR2,
    p_new_name           IN VARCHAR2 DEFAULT NULL,
    p_data_source_type   IN VARCHAR2,
    p_table              IN VARCHAR2 DEFAULT NULL,
    p_preserve_case      IN VARCHAR2 DEFAULT 'N',
    p_sql_query          IN VARCHAR2 DEFAULT NULL,
    p_where_clause        IN VARCHAR2 DEFAULT NULL,
    p_inline_data         IN CLOB     DEFAULT NULL,
    p_order_by_column     IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_blueprint	Identifies the blueprint.
p_name	Name of a data source. Name is upper cased and must be 26 characters or less.
p_new_name	New name of a data source (rename). Name is upper cased and must be 26 characters or less.
p_data_source_type	TABLE, SQL_QUERY.
p_table	For source type = TABLE. Typically this matches p_name.
p_preserve_case	Defaults to N which forces p_table_name to uppercase. If Y, preserves casing of p_table.
p_sql_query	For p_data_source_type = SQL_QUERY.
p_where_clause	For p_data_source_type = TABLE, this adds the where clause. Do not include "where" keyword (for example deptno <= 20).
p_inline_data	Used for p_data_source_type = JSON_DATA.
p_order_by_column	Not used.

Example

```
BEGIN
    apex_dg_data_gen.update_data_source(
        p_blueprint      => 'Cars',
        p_name           => 'apex_dg_builtin_cars',
        p_data_source_type => 'TABLE',
        p_table          => 'apex_dg_builtin_cars');
END;
```

24.26 UPDATE_TABLE Procedure

This procedure updates the attributes for a blueprint table. The logical key is `p_blueprint` and `p_table_name`.

Syntax

```
APEX_DG_DATA_GEN.UPDATE_TABLE (
    p_blueprint      IN VARCHAR2,
    p_table_name     IN VARCHAR2,
    p_new_table_name IN VARCHAR2      DEFAULT NULL,
    p_sequence       IN PLS_INTEGER,
    p_preserve_case  IN VARCHAR2      DEFAULT 'N',
    p_display_name   IN VARCHAR2      DEFAULT NULL,
    p_singular_name  IN VARCHAR2      DEFAULT NULL,
    p_plural_name    IN VARCHAR2      DEFAULT NULL,
    p_rows           IN NUMBER        DEFAULT 0,
    p_max_rows       IN VARCHAR2      DEFAULT NULL )
```

Parameters

Parameter	Description
<code>p_blueprint</code>	Identifier for the blueprint.
<code>p_table_name</code>	Name of table that can exist or not exist.
<code>p_new_table_name</code>	New table name (rename).
<code>p_sequence</code>	1 for first table, 2 for second, and so forth.
<code>p_preserve_case</code>	Defaults to N which forces <code>p_new_table_name</code> to uppercase. If Y, preserves casing of <code>p_new_table_name</code> .
<code>p_display_name</code>	Friendly display name.
<code>p_singular_name</code>	Singluar friendly name.
<code>p_plural_name</code>	Plural friendly name.
<code>p_rows</code>	Number of rows to generate for this table.
<code>p_max_rows</code>	If NULL, then <code>p_rows</code> determines the number of rows, otherwise random rows between <code>p_rows</code> and <code>p_max_rows</code> are used when generating output.

Example

```
BEGIN
    apex_dg_data_gen.update_table(
        p_blueprint      => 'Cars',
        p_table_name     => 'MY_CARS',
        p_sequence       => 20,
        p_new_table_name => 'MY_NEW_CARS',
        p_display_name   => 'My great cars 2',
        p_singular_name  => 'My car',
        p_plural_name    => 'My Cars',
        p_rows           => '50',
    );
END;
```

```

BEGIN
    apex_dg_data_gen.update_table(
        p_blueprint          => 'Cars',
        p_table_name         => 'my_cars',
        p_sequence           => 10,
        p_rows               => '50',
        p_use_existing_table => 'Y',
    );
END;

```

24.27 VALIDATE_BLUEPRINT Procedure

This procedure validates the blueprint by checking the validity of the generated SQL.

Syntax

```
APEX_DG_DATA_GEN.VALIDATE_BLUEPRINT (
    p_blueprint      IN      VARCHAR2,
    p_format         IN      VARCHAR2,
    p_errors         OUT     CLOB )
```

Parameters

Parameter	Description
p_blueprint	Name of the blueprint.
p_format	CSV, SQL INSERT, JSON, PRETTY JSON, INSERT INTO, or FAST INSERT INTO.
p_errors	Clob holds error output.

Example

```

DECLARE
    l_errors    clob;
BEGIN
    apex_dg_output.validate_blueprint
        (p_blueprint      => 'Cars',
         p_format         => 'JSON',
         p_errors         => l_errors
        );
END;

```

24.28 VALIDATE_INSTANCE_SETTING Procedure

This procedure validates appropriate instance settings (table, column, generation level).

Syntax

```
APEX_DG_DATA_GEN.VALIDATE_INSTANCE_SETTING (
    p_json       IN      CLOB,
```

```
p_valid      OUT NOCOPY  VARCHAR2,  
p_message   OUT NOCOPY  CLOB )
```

Parameters

Parameter	Description
p_json	JSON Document.
p_valid	Out parameter to identify whether settings are valid.
p_result	Out parameter with a detailed message.

Example

```
DECLARE  
    l_is_valid varchar2(30);  
    l_message clob;  
BEGIN  
    apex_dg_data_gen.validate_instance_setting(  
        p_json          => '<json_doc>',  
        p_valid         => l_is_valid,  
        p_message       => l_message);  
END;
```

25

APEX_ERROR

The APEX_ERROR package provides the interface declarations and utility functions for an error handling function and includes procedures and functions to raise errors in an APEX 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](#)
- [AUTO_SET_ASSOCIATED_ITEM Procedure](#)
- [EXTRACT_CONSTRAINT_NAME Function](#)
- [GET_FIRST_ORA_ERROR_TEXT Function](#)
- [HAVE_ERRORS_OCCURRED Function](#)
- [INIT_ERROR_RESULT Function](#)

25.1 Constants and Attributes Used for Result Types

The following constants are used for the API parameter `p_display_location` and the attribute `display_location` in the `t_error` and `t_error_result` types.

```
c_inline_with_field           constant varchar2(40):='INLINE_WITH_FIELD';
c_inline_with_field_and_notif constant
varchar2(40):='INLINE_WITH_FIELD_AND_NOTIFICATION';
c_inline_in_notification      constant
varchar2(40):='INLINE_IN_NOTIFICATION';
c_on_error_page               constant varchar2(40):='ON_ERROR_PAGE';
```

The following constants are used for the API parameter `associated_type` in the `t_error` type.

```
c_ass_type_page_item          constant varchar2(30):='PAGE_ITEM';
c_ass_type_region              constant varchar2(30):='REGION';
c_ass_type_region_column       constant varchar2(30):='REGION_COLUMN';
```

The following record structure is passed into an error handling callout function and contains all the relevant information about the error.

```
type t_error is record (
    message                varchar2(32767),
```

```

        /* Error message which will be displayed */
additional_info          varchar2(32767),
        /* Only used for display_location ON_ERROR_PAGE to display additional
error information */
display_location         varchar2(40),
        /* Use constants "used for display_location" below */
association_type         varchar2(40),
        /* Use constants "used for 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 alias */
row_num                  pls_integer,
        /* Associated tabular form row */
apex_error_code           varchar2(255),
        /* Contains the system message code if it's an error raised by APEX */
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 */
is_common_runtime_error   boolean,
        /* TRUE for internal authorization, session and session state errors
that normally should not be masked
        by an error handler */
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_application.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
name */
);

```

```
alias */
);
```

25.2 Example of an Error Handling Function

The following is an example of an error handling function.

```
create or replace function apex_error_handling_example (
    p_error in apex_error.t_error )
    return apex_error.t_error_result
IS
    l_result          apex_error.t_error_result;
    l_reference_id   number;
    l_constraint_name varchar2(255);
BEGIN
    l_result := apex_error.init_error_result (
        p_error => p_error );

    -- If it's an internal error raised by APEX, like an invalid statement or
    -- code which can't be executed, the error text might contain security
    -- sensitive information. To avoid this security problem we can rewrite
    the
    -- error to a generic error message and log the original error message for
    -- further investigation by the help desk.

    IF p_error.is_internal_error THEN

        -- mask all errors that are not common runtime errors (Access Denied
        -- errors raised by application / page authorization and all errors
        -- regarding session and session state)

        IF not p_error.is_common_runtime_error THEN

            -- log error for example with an autonomous transaction and return
            -- l_reference_id as reference#
            -- l_reference_id := log_error (
            --                     p_error => p_error );

            -- Change the message to the generic error message which doesn't
            -- expose any sensitive information.

            l_result.message := 'An unexpected internal application
                                error has occurred. ' ||
                                'Please get in contact with XXX and provide
                                'reference#
                                '||to_char(l_reference_id,
                                '999G999G999G990') ||
                                ' for further investigation.';

            l_result.additional_info := null;
        END IF;
    ELSE

        -- 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;
```

25.3 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 Oracle APEX 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

Parameters	Description
<code>p_message</code>	Displayed error message.
<code>p_additional_info</code>	Additional error information needed if the error is displayed on the error page.
<code>p_display_location</code>	Specifies where the error message is displayed. Use the constant <code>apex_error.c_inline_in_notification</code> or <code>apex_error.c_on_error_page</code> . See Constants and Attributes Used for Result Types .

Example

This example adds a custom error message to the error stack. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_message      => 'This custom account is not active!',
    p_display_location => apex_error.c_inline_in_notification );
```

25.4 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 APEX application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_message          IN VARCHAR2,
    p_additional_info IN VARCHAR2 DEFAULT NULL,
    p_display_location IN VARCHAR2,
    p_page_item_name  IN VARCHAR2 )
```

Parameters

Parameters	Description
<code>p_message</code>	Displayed error message.
<code>p_additional_info</code>	Additional error information needed if the error is displayed on the error page.
<code>p_display_location</code>	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 Constants and Attributes Used for Result Types .
<code>p_page_item_name</code>	Name of the page item on the current page 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 the display location.

Example

This example adds a custom error message to the error stack. The `P5_CUSTOMER_ID` item is highlighted on the page. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_message      => 'Invalid Customer ID!',
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_page_item_name  => 'P5_CUSTOMER_ID');
```

25.5 ADD_ERROR Procedure Signature 3

Adds an error message to the error stack based on a text message defined in Shared Components. This error message can be displayed in 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 Oracle APEX application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_error_code          IN VARCHAR2,
    p0                   IN VARCHAR2 DEFAULT NULL,
    p1                   IN VARCHAR2 DEFAULT NULL,
    p2                   IN VARCHAR2 DEFAULT NULL,
    p3                   IN VARCHAR2 DEFAULT NULL,
    p4                   IN VARCHAR2 DEFAULT NULL,
    p5                   IN VARCHAR2 DEFAULT NULL,
    p6                   IN VARCHAR2 DEFAULT NULL,
    p7                   IN VARCHAR2 DEFAULT NULL,
    p8                   IN VARCHAR2 DEFAULT NULL,
    p9                   IN VARCHAR2 DEFAULT NULL,
    p_escape_placeholders IN BOOLEAN DEFAULT TRUE,
    p_additional_info     IN VARCHAR2 DEFAULT NULL,
    p_display_location   IN VARCHAR2,
    p_page_item_name     IN VARCHAR2 );
```

Parameters

Parameters	Description
<code>p_error_code</code>	Name of shared component text message.
<code>p_additional_info</code>	Additional error information needed if the error is displayed on the error page.
<code>p0 through p9</code>	Values for %0 through %9 placeholders defined in the text message.
<code>p_escape_placeholders</code>	If set to TRUE, the values provided in <code>p0</code> through <code>p9</code> are escaped with <code>sys.htf.escape_sc</code> before replacing the placeholder in the text message. If set to FALSE, values are not escaped.
<code>p_display_location</code>	Specifies where the error message is displayed. Use the constants defined for <code>p_display_location</code> . See Constants and Attributes Used for Result Types .
<code>p_page_item_name</code>	Name of the page item on the current page 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 the display location.

Example

This example adds 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' );
```

25.6 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 Oracle APEX application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of `apex_error.c_on_error_page`.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_message          IN VARCHAR2,
    p_additional_info  IN VARCHAR2 DEFAULT NULL,
    p_display_location IN VARCHAR2,
    p_region_id       IN NUMBER,
    p_column_alias    IN VARCHAR2 DEFAULT NULL,
    p_row_num         IN NUMBER );
```

Parameters

Parameters	Description
p_message	Displayed error message.
p_additional_info	Additional error information needed if the error is displayed on the error page.
p_display_location	Specifies where the error message is displayed. Use the constant <code>apex_error.c_inline_with_field</code> or <code>apex_error.c_inline_with_field_and_notif</code> . See Constants and Attributes Used for Result Types .
p_region_id	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> .

Parameters	Description
p_column_alias	Name of a tabular form column alias defined for p_region_id that is highlighted if apex_error.c_inline_with_field or apex_error.c_inline_with_field_and_notif are used as a display location.
p_row_num	Number of the tabular form row where the error occurred.

Example

This example adds a custom error message for a tabular form, where the column CUSTOMER_ID is highlighted, to the error stack. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_message          => 'Invalid Customer ID!',
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_region_id        => l_region_id,
    p_column_alias     => 'CUSTOMER_ID',
    p_row_num          => l_row_num );
```

25.7 ADD_ERROR Procedure Signature 5

Adds an error message to the error stack of a tabular form based on a text message defined in Shared Components. This error message can be displayed in 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 Oracle APEX application has performed the last validation or process. Otherwise, the error is ignored if it does not have a display location of apex_error.c_on_error_page.

Syntax

```
APEX_ERROR.ADD_ERROR (
    p_error_code      IN VARCHAR2,
    p0                IN VARCHAR2 DEFAULT NULL,
    p1                IN VARCHAR2 DEFAULT NULL,
    p2                IN VARCHAR2 DEFAULT NULL,
    p3                IN VARCHAR2 DEFAULT NULL,
    p4                IN VARCHAR2 DEFAULT NULL,
    p5                IN VARCHAR2 DEFAULT NULL,
    p6                IN VARCHAR2 DEFAULT NULL,
    p7                IN VARCHAR2 DEFAULT NULL,
    p8                IN VARCHAR2 DEFAULT NULL,
    p9                IN VARCHAR2 DEFAULT NULL,
    p_escape_placeholders IN BOOLEAN DEFAULT TRUE,
    p_additional_info   IN VARCHAR2 DEFAULT NULL,
    p_display_location  IN VARCHAR2,
    p_region_id        IN NUMBER,
```

```
p_column_alias      IN VARCHAR2 DEFAULT NULL,
p_row_num          IN NUMBER );
```

Parameters

Parameters	Description
p_error_code	Name of shared component text message.
p0 through p9	Values for %0 through %9 placeholders defined in the text message.
p_escape_placeholders	If set to TRUE, the values provided in p0 through p9 are escaped with sys.htf.escape_sc before replacing the placeholder in the text message. If set to FALSE, values are not escaped.
p_additional_info	Additional error information needed if the error is displayed on the error page.
p_display_location	Specifies where the error message is displayed. Use the constants defined for p_display_location. See Constants and Attributes Used for Result Types .
p_region_id	The ID of the tabular form region on the current page. The ID can be read from the view APEX_APPLICATION_PAGE_REGIONS.
p_column_alias	The name of the tabular form column alias defined for p_region_id that is highlighted if apex_error.c_inline_with_field or apex_error.c_inline_with_field_and_notif are used as a display location.
p_row_num	Number of the tabular form row where the error occurred.

Example

This example adds a custom error message, where the text is stored in a text message, to the error stack. The CUSTOMER_ID column on the tabular form is highlighted. The error message is displayed inline in a notification. This example can be used in a validation or process.

```
apex_error.add_error (
    p_error_code      => 'INVALID_CUSTOMER_ID',
    p0                 => l_customer_id,
    p_display_location => apex_error.c_inline_with_field_and_notif,
    p_region_id       => l_region_id,
    p_column_alias    => 'CUSTOMER_ID',
    p_row_num         => l_row_num );
```

25.8 AUTO_SET_ASSOCIATED_ITEM Procedure

This procedure automatically sets the associated page item or tabular form column based on a constraint contained in p_error.ora_sqlerrm. This procedure performs the following:

- Identifies the constraint by searching for the schema.constraint pattern.
- Only supports constraints of type P, U, R and C.
- For constraints of type C (check constraints), the procedure parses the expression to identify those columns that are used in the constraints expression.
- Using those columns, the procedure gets the first visible page item or tabular form column that is based on that column and set it as associated p_error_result.page_item_name or p_error_result.column_alias.

- If a page item or tabular form column was found, `p_error_result.display_location` is set to `apex_error.c_inline_with_field_and_notif`.

Syntax

```
APEX_ERROR.AUTO_SET_ASSOCIATED_ITEM (
    p_error_result IN OUT NOCOPY t_error_result,
    p_error         IN                 t_error );
```

Parameters

Parameters	Description
<code>p_error_result</code>	The result variable of your error handling function.
<code>p_error</code>	The <code>p_error</code> parameter of your error handling function.

Example

See an example of how to use this procedure in [Example of an Error Handling Function](#).

25.9 EXTRACT_CONSTRAINT_NAME Function

This function extracts a constraint name contained in `p_error.ora_sqlerrm`. The constraint must match the pattern `schema.constraint`.

Syntax

```
APEX_ERROR.EXTRACT_CONSTRAINT_NAME (
    p_error           IN t_error,
    p_include_schema IN BOOLEAN DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameters	Description
<code>p_error</code>	The <code>p_error</code> parameter of your error handling function.
<code>p_include_schema</code>	If set to TRUE, the result is prefixed with the schema name. For example, <code>HR.DEMO_PRODUCT_INFO_PK</code> . If set to FALSE, only the constraint name is returned.

Example

See an example of how to use this function in [Example of an Error Handling Function](#).

25.10 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

Parameters	Description
p_error	The p_error parameter of your error handling function.
p_include_error_no	If set to TRUE, ORA-xxxx is included in the returned error message. If set to FALSE, only the error message text is returned.

Example

See an example of how to use this function in [Example of an Error Handling Function](#).

25.11 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.

```
IF NOT apex_error.have_errors_occurred THEN
    ...
END IF;
```

25.12 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

Parameters	Description
p_error	The p_error parameter of your error handling function.

Example

See an example of how to use this function in [Example of an Error Handling Function](#).

APEX_ESCAPE

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

- Constants
- CSS_SELECTOR Function
- CSV Function Signature 1
- CSV Function Signature 2
- GET_CSV_ENCLOSED_BY Function
- GET_CSV_SEPARATED_BY Function
- HTML Function
- HTML_ALLOWLIST Function
- HTML_ALLOWLIST_CLOB Function
- HTML_ATTRIBUTE Function
- HTML_ATTRIBUTE_CLOB Function
- HTML_CLOB Function
- HTML_TRUNC Function Signature 1
- HTML_TRUNC Function Signature 2
- JS_LITERAL Function
- JS_LITERAL_CLOB Function
- JSON Function
- JSON_CLOB Function
- LDAP_DN Function
- LDAP_SEARCH_FILTER Function
- NOOP Function Signature 1
- NOOP Function Signature 2
- REGEXP Function
- SET_CSV_PARAMETERS Procedure
- SET_HTML_ESCAPING_MODE Procedure
- STRIPHTML Function Signature 1
- STRIPHTML Function Signature 2

26.1 Constants

The APEX_ESCAPE package uses the following constants.

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

26.2 CSS_SELECTOR Function

This function escapes meta-characters in a string used in a CSS selector.

See <http://api.jquery.com/category/selectors/> for a list of characters.

Syntax

```
APEX_ESCAPE.CSS_SELECTOR (
    p_string      IN VARCHAR2 )
RETURN VARCHAR2 deterministic;
```

Parameters

Parameter	Description
p_string	The string to be escaped.

Example

The following example ensures that the meta-character @ in mary@example.com is escaped and ignored by jQuery.

```
DECLARE
    l_name varchar2(30) := 'mary@example.com';
BEGIN
    apex_javascript.add_onload_code( '$('#' ||
apex_escape.js_literal( apex_escape.css_selector( l_name ), null ) ||
'"').hide();' );
END;
```

26.3 CSV Function Signature 1

This function escapes special characters in a CSV value (VARCHAR2).

Syntax

```
APEX_ESCAPE.CSV (
    p_string      IN VARCHAR2,
    p_quote       IN BOOLEAN DEFAULT TRUE,
```

```
p_strip_html      IN BOOLEAN  DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	The string to be escaped.
p_quote	If TRUE (default) and p_string contains special characters, enclose the result with the p_enclose_by parameter of set_csv_parameters.
p_strip_html	Default FALSE. If TRUE, remove any HTML tags.

Example

The following example prints a CSV report with employee IDs and names and non-default ; as separator.

```
BEGIN
    apex_escape.set_csv_parameters (
        p_enclosed_by => '',
        p_separated_by => ';');

    for i in ( select empno, ename from emp ) loop
        sys.dbms_output.put_line (
            i.empno || ';' || apex_escape.csv(i.ename) );
    END loop;
END;
```

See Also:

- [CSV Function Signature 2](#)
- [SET_CSV_PARAMETERS Procedure](#)
- [GET_CSV_ENCLOSED_BY Function](#)
- [GET_CSV_SEPARATED_BY Function](#)

26.4 CSV Function Signature 2

This function escapes special characters in a CSV value (CLOB).

Syntax

```
APEX_ESCAPE.CSV (
    p_string      IN CLOB,
    p_quote       IN BOOLEAN  DEFAULT TRUE,
```

```
p_strip_html      IN BOOLEAN  DEFAULT FALSE )
RETURN CLOB;
```

Parameters

Parameter	Description
p_string	The string to be escaped.
p_quote	If TRUE (default) and p_string contains special characters, enclose the result with the p_enclose_by parameter of set_csv_parameters.
p_strip_html	Default FALSE. If TRUE, remove any HTML tags.

Example

The following example prints a CSV report with employee IDs and bio (a CLOB column) and non-default ; as separator.

```
BEGIN
    apex_escape.set_csv_parameters (
        p_enclosed_by => "'",
        p_separated_by => ';');
    for i in ( select empno, bio from emp ) loop
        sys.dbms_output.put_line (
            i.empno || ';' || apex_escape.csv(i.bio) );
    END loop;
END;
```

See Also:

- [CSV Function Signature 1](#)
- [SET_CSV_PARAMETERS Procedure](#)
- [GET_CSV_ENCLOSED_BY Function](#)
- [GET_CSV_SEPARATED_BY Function](#)

26.5 GET_CSV_ENCLOSED_BY Function

This function returns the CSV enclose by character.

Syntax

```
APEX_ESCAPE.GET_CSV_ENCLOSED_BY
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
None.	None.

 **See Also:**

- [CSV Function Signature 1](#)
- [CSV Function Signature 2](#)
- [SET_CSV_PARAMETERS Procedure](#)
- [GET_CSV_SEPARATED_BY Function](#)

26.6 GET_CSV_SEPARATED_BY Function

This function returns the CSV separated by character.

Syntax

```
APEX_ESCAPE.GET_CSV_SEPARATED_BY  
RETURN VARCHAR2;
```

Parameters

Parameter	Description
None.	None.

 **See Also:**

- [CSV Function Signature 1](#)
- [CSV Function Signature 2](#)
- [SET_CSV_PARAMETERS Procedure](#)
- [GET_CSV_ENCLOSED_BY Function](#)

26.7 HTML Function

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

This function's result depends on the escaping mode that is defined by using `apex_escape.set_html_escaping_mode`. By default, the escaping mode is Extended, but it can

be overridden by manually calling `set_htmlEscapingMode` 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, depicts ASCII characters that the function transforms and their escaped values:

Raw ASCII Characters	Returned Escaped Characters
&	&
"	"
<	<
>	>
'	'
/	/

Syntax

```
APEX_ESCAPE.HTML (
    p_string IN VARCHAR2 )
    RETURN VARCHAR2 deterministic;
```

Parameters

Parameter	Description
<code>p_string</code>	The string text that is escaped.

Example

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

```
DECLARE
procedure eq(p_str1 in varchar2,p_str2 in varchar2)
    is
BEGIN
    IF p_str1||'.' <> p_str2||'.' THEN
        raise_application_error(-20001,p_str1||' '<>'||p_str2);
    END IF;
END eq;
BEGIN
    apex_escape.set_htmlEscapingMode('B');
    eq(apex_escape.html('hello &"<>''/'), 'hello &quot;&lt;&gt;''/');
    apex_escape.set_htmlEscapingMode('E');
    eq(apex_escape.html('hello &"<>''/'), 'hello
&quot;&lt;&gt;&#x27;&#x2F;');
END;
```

 **See Also:**

- [SET_HTML_ESCAPING_MODE Procedure](#)

26.8 HTML_ALLOWLIST Function

The `HTML_ALLOWLIST` function performs HTML escape on all characters in the input text except the specified allowlist 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_ALLOWLIST (
    p_html          IN VARCHAR2,
    p_allowlist_tags IN VARCHAR2 DEFAULT c_html_allowlist_tags )
    RETURN VARCHAR2 deterministic;
```

Parameters

Parameter	Description
<code>p_html</code>	The text string that is filtered.
<code>p_allowlist_tags</code>	The comma separated list of tags that stays in <code>p_html</code> .

Example

This example shows how to use `HTML_ALLOWLIST` to remove unwanted html markup from a string, while preserving allowlisted tags.

```
BEGIN
    sys.hpt.p(apex_escape.html_allowlist(
        '<h1>Hello<script>alert("XSS");</script></h1>'));
END;
```

 **See Also:**

- [SET_HTML_ESCAPING_MODE Procedure](#)

26.9 HTML_ALLOWLIST_CLOB Function

This function performs HTML escape on all characters in the input text except the specified allowlist 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_ALLOWLIST_CLOB (
    p_html          IN CLOB,
    p_allowlist_tags IN VARCHAR2 DEFAULT c_html_allowlist_tags )
RETURN CLOB deterministic;
```

Parameters

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

See Also:

- [HTML_ALLOWLIST Function](#)
- [HTML_CLOB Function](#)
- [HTML_TRUNC Function Signature 2](#)
- [HTML_ATTRIBUTE_CLOB Function](#)
- [SET_HTML_ESCAPING_MODE Procedure](#)

26.10 HTML_ATTRIBUTE Function

! Important:

When using HTML_ATTRIBUTE for plain text attributes (such as *title*, *placeholder*, *aria-label*), you may expose HTML code to end users. To exclude HTML code exposed to end users for similar plain text attributes, avoid calls to HTML_ATTRIBUTE function.

⚠ WARNING:

Do not use the HTML_ATTRIBUTE function to escape such attributes as *aria-label*, *alt*, *summary* and other attributes because they produce visually hidden content that is not obvious when HTML code is exposed to users of assistive technologies.

Tip:

Oracle recommends [GET_HTML_ATTR Function](#) to **escape all HTML attributes** instead of this function.

GET_HTML_ATTR enables you to choose the proper algorithm to escape the attribute value.

This function escapes the values of HTML entity attributes. The API hex escapes everything that is *not* alphanumeric or within one of the following characters:

- ,
- .
- -
- _

Syntax

```
APEX_ESCAPE.HTML_ATTRIBUTE (
    p_string IN VARCHAR2 )
    RETURN VARCHAR2 deterministic;
```

Parameters

Parameter	Description
p_string	The text string that is escaped.

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
    http.p('<ul>');
    for l_data in ( select title, cls, body
                    from my_topics )
    LOOP
        sys.http.p('<li><span class=""||'
                   apex_escape.html_attribute(l_data.cls)||'">' ||
                   apex_escape.html(l_data.title)||'</span>');
        sys.http.p(apex_escape.html_trunc(l_data.body));
        sys.http.p('</li>');
    END LOOP;

    http.p('</ul>');
END;
```

 **See Also:**

- [GET_HTML_ATTR Function](#)
- [SET_HTML_ESCAPING_MODE Procedure](#)

26.11 HTML_ATTRIBUTE_CLOB Function

This function escapes the values of HTML entity attributes. It hex escapes everything that is *not* alphanumeric or in one of the following characters:

- ,
- .
- -
- _

Syntax

```
APEX_ESCAPE.HTML_ATTRIBUTE_CLOB (
    p_string      IN CLOB )
    RETURN CLOB deterministic;
```

Parameters

Parameter	Description
p_string	The text string that is escaped.

 **See Also:**

- [HTML_ALLOWLIST Function](#)
- [HTML_CLOB Function](#)
- [HTML_TRUNC Function Signature 2](#)
- [HTML_ALLOWLIST_CLOB Function](#)
- [SET_HTML_ESCAPING_MODE Procedure](#)

26.12 HTML_CLOB 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 `SET_HTML_ESCAPING_MODE`. By default, the escaping mode is "Extended", but it can be overridden by manually calling `SET_HTML_ESCAPING_MODE` or by setting the "application security

attribute HTML Escaping Mode" to "Basic." If the mode is Basic, the function behaves like SYS.HTF.ESCAPE_SC. Otherwise, the rules below apply.

The following table, depicts ASCII characters that the function transforms and their escaped values:

Table 26-1 Escaped Values for Transformed ASCII Characters

Raw ASCII Characters	Returned Escaped Characters
&	&
"	"
<	<
>	>
'	'
/	/

In addition, the function may escape unicode characters if the database NLS character set is *not* UTF-8 or if the REQUESTIANA_CHARSET HTTP header variable is set to something different than UTF-8 (which is the default). If unicode escaping applies, these characters are escaped via &#xHHHH; where HHHH is the unicode hex code.

Syntax

```
APEX_ESCAPE.HTML_CLOB (
    p_string      IN CLOB )
    RETURN CLOB deterministic;
```

Parameters

Parameter	Description
p_string	The string text that is escaped.

Example

The following example tests escaping in basic (B) and extended (E) mode.

```
DECLARE
    procedure eq(p_str1 in clob,p_str2 in clob)
    is
    BEGIN
        IF dbms_lob.compare(p_str1||'.', p_str2||'.') <> 0 THEN
            raise_application_error(-20001,'p_str1 <> p_str2');
        END IF;
    END eq;
BEGIN
    apex_escape.set_htmlEscapingMode('B');
    eq(apex_escape.html_clob('hello &"<>'/''), 'hello &quot;&lt;&gt;''/');
    apex_escape.set_htmlEscapingMode('E');
    eq(apex_escape.html_clob('hello &"<>'/''), 'hello
&amp;quot;&lt;&gt;';#x27;#x2F;');
END;
```

 See Also:

- [HTML Function](#)
- [HTML_TRUNC Function Signature 2](#)
- [HTML_ALLOWLIST_CLOB Function](#)
- [HTML_ATTRIBUTE_CLOB Function](#)
- [SET_HTML_ESCAPING_MODE Procedure](#)

26.13 HTML_TRUNC Function Signature 1

This function escapes HTML and limits the returned string to `p_length` bytes. This function returns the first `p_length` bytes of an input VARCHAR2 and escapes them. You can use this function if the input VARCHAR2 is 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 VARCHAR2,
    p_length      IN NUMBER      DEFAULT 4000 )
RETURN VARCHAR2;
```

Parameters

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

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
    http.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;
    http.p('</ul>');
END;
```

 See Also:

- [SET_HTML_ESCAPING_MODE Procedure](#)

26.14 HTML_TRUNC Function Signature 2

This 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 is 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 deterministic;
```

Parameters

Parameter	Description
<code>p_string</code>	The text string to be escaped (CLOB).
<code>p_length</code>	The number of bytes from <code>p_string</code> that are escaped. For ASCII characters, a byte is a character. For Unicode characters, a single character can take up to 4 bytes.

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

 See Also:

- [HTML_TRUNC Function Signature 1](#)
- [HTML_CLOB Function](#)
- [HTML_ALLOWLIST_CLOB Function](#)
- [HTML_ATTRIBUTE_CLOB Function](#)
- [SET_HTML_ESCAPING_MODE Procedure](#)

26.15 JS_LITERAL Function

The `JS_LITERAL` function escapes and optionally enquotes a JavaScript string. This function replaces non-immune characters with `\xHH` or `\uHHHH` equivalents. The result can be injected into JavaScript code, within `<script>` tags or inline (`javascript:nnn`). Immune characters include:

- a through z
- A through Z
- 0 through 9
- commas ,
- periods .
- underscores _

If the output should not be enclosed in quotes, then the parameter `p_quote` is `NULL`.

If `p_quote` has a value, printable ASCII 7 characters are not escaped except for & < > `` `/ \ %

Syntax

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

Parameters

Parameter	Description
<code>p_string</code>	The text string that is escaped.
<code>p_quote</code>	If not <code>NULL</code> , this string is placed on the left and right of the result. The quotation character must be a single- or double-quotation mark.

Example

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

```
DECLARE
    l_string varchar2(4000) := 'O''Brien';
BEGIN
```

```

        sys.htp.p('<script>' ||
           'alert('||apex_escape.js_literal(l_string)||');'||'</script>');
END;

```

26.16 JS_LITERAL_CLOB Function

This function escapes and optionally enquotes a JavaScript string. This function replaces non-immune characters with \xHH or \uHHHH equivalents. The result can be injected into JavaScript code, within `<script>` tags or inline (`javascript:nnn`). Immune characters include:

- a through z
- A through Z
- 0 through 9
- commas ,
- periods .
- underscores _

If the output should not be enclosed in quotes, then the parameter `p_quote` is NULL.

If `p_quote` has a value, printable ASCII 7 characters are not escaped except for & < > `` / \ %

Syntax

```
APEX_ESCAPE.JS_LITERAL_CLOB (
    p_string    IN CLOB )
RETURN CLOB;
```

Parameters

Parameter	Description
<code>p_string</code>	The text string that is escaped.
<code>p_quote</code>	If not NULL, this string is placed on the left and right of the result. The quotation character must be a single- or double-quotation mark.

Example

The following example describes how to use `JS_LITERAL` to escape special characters in the `l_string` variable.

```

DECLARE
    l_string clob := 'O''Brien';
BEGIN
    sys.htp.p(
        to_clob('<script>') ||
        'alert('|| apex_escape.js_literal_clob(l_string) || ');'||'</script>');
END;

```

26.17 JSON Function

This function returns `p_string` with all special characters escaped.

Syntax

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

Parameters

Parameter	Description
<code>p_string</code>	The string to be escaped.

Returns

Return	Description
<code>VARCHAR2</code>	The escaped string.

Example

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

```
DECLARE
    l_string varchar2(4000) := 'O''Brien';
BEGIN
    sys.htp.p('{ "name": "'||apex_escape.json(l_string)||'"}');
END;
```

26.18 JSON_CLOB Function

This function returns `p_string` with all special characters escaped.

Syntax

```
APEX_ESCAPE.JSON_CLOB (
    p_string      IN CLOB )
RETURN CLOB;
```

Parameters

Parameter	Description
<code>p_string</code>	The string to be escaped.

Example

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

```
DECLARE
    l_string clob := 'O''Brien';
BEGIN
    sys.htp.p('{ "name": "' || apex_escape.json_clob(l_string) || '"}');
END;
```

26.19 LDAP_DN Function

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

Syntax

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

Parameters

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

Example

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

```
DECLARE
    l_name varchar2(4000) := 'Joe+User';
    l_escaped varchar2(4000);
BEGIN
    l_escaped := apex_escape.ldap_dn(l_name);
    htp.p(l_name||' becomes'||l_escaped);
END;
```

 Note:[LDAP_SEARCH_FILTER Function](#)

26.20 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

Parameter	Description
<code>p_string</code>	The text string that is escaped.
<code>p_reserved_chars</code>	A list of characters that when found in <code>p_string</code> is escaped with <code>\xx</code> where <code>xx</code> is the character's ASCII hexadecimal code.
<code>p_escape_non_ascii</code>	If TRUE, characters above ascii 127 in <code>p_string</code> are escaped with <code>\xx</code> where <code>xx</code> is the character's ASCII hexadecimal code. This is supported by RFCs 4514, but may cause errors with older LDAP servers and Microsoft AD.

Example

This example escapes the text in `l_name` and places the result in `l_escaped`.

```
DECLARE
  l_name varchar2(4000) := 'Joe*User';
  l_escaped varchar2(4000);
BEGIN
  l_escaped := apex_escape.ldap_search_filter(l_name);
  http.p(l_name || ' becomes ' || l_escaped);
END;
```

 Note:[LDAP_DN Function](#)

26.21 NOOP Function Signature 1

This function returns `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

Parameter	Description
<code>p_string</code>	The input text string.

Example

This example shows how to use `NOOP` to show the developer's intention to explicitly not escape text.

```
BEGIN
    sys.hpt.p(apex_escape.noop('Cats & Dogs'));
END;
```

26.22 NOOP Function Signature 2

This function returns `p_string` (CLOB) 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 CLOB )
    RETURN CLOB deterministic;
```

Parameters

Parameter	Description
<code>p_string</code>	The input text string.

Example

The following example shows how to use `NOOP` to show the developer's intention to explicitly *not* escape text.

```
BEGIN
    sys.hpt.p(apex_escape.noop( to_clob('Cats & Dogs') ));
END;
```

26.23 REGEXP Function

This function escapes characters that can change the context in a regular expression. It should be used to secure user input. The following list depicts ascii characters that the function escapes with a backslash (\):

```
\.^$*+-?()
```

Syntax

```
APEX_ESCAPE.REGEXP (
    p_string      IN VARCHAR2 )
```

Parameters

Parameter	Description
p_string	Text to escape.

Example

The following example ensures the special character "-" in "Mary-Ann" is 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')
        END IF;
END
```

26.24 SET_CSV_PARAMETERS Procedure

This procedure sets parameters for the CSV function.

Syntax

```
APEX_ESCAPE.SET_CSV_PARAMETERS (
    p_enclosed_by      IN VARCHAR2 DEFAULT NULL,
    p_separated_by     IN VARCHAR2 DEFAULT NULL,
    p_escape_formulas IN BOOLEAN  DEFAULT NULL );
```

Parameters

Parameter	Description
p_enclosed_by	The string to enclose CSV values. If NULL (default), fall back to double quote.
p_separated_by	The string to separate CSV values. If NULL (default), determine the separator by checking the NLS decimal separator. If that is comma (,) the CSV separator is semicolon (;) otherwise it is comma (,).
p_escape_formulas	Default TRUE, but can be overridden with instance parameter CSV_DOWNLOAD_ESCAPE_FORMULAS If TRUE, escape formula cells by prepending them with a space. Formula cells can start with: <ul style="list-style-type: none"> • = • @ • + • - The sign characters are only escaped if they are not part of numbers.

 **See Also:**

- [CSV Function Signature 1](#)
- [CSV Function Signature 2](#)
- [GET_CSV_ENCLOSED_BY Function](#)
- [GET_CSV_SEPARATED_BY Function](#)

26.25 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

Parameter	Description
p_mode	If B, then do basic escaping, like sys.htf.escape_sc. If E, then do extended escaping.

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_htmlEscapingMode('B');
    eq(apex_escape.html('hello &"<>''), 'hello &quot;&lt;&gt;'');
    apex_escape.set_htmlEscapingMode('E');
    eq(apex_escape.html('hello &"<>''), 'hello
&quot;&lt;&gt;';');
END;

```

See Also:

- [HTML Function](#)
- [HTML_ALLOWLIST Function](#)
- [HTML_ATTRIBUTE Function](#)
- [HTML_TRUNC Function Signature 1](#)

26.26 STRIPHTML Function Signature 1

This function returns `p_string` (VARCHAR2) removing HTML tags, leaving plain text.

This function removes all HTML attributes regardless of the type of HTML content. For example, it preserves content such as JavaScript and CSS, but removes script and CSS HTML tags.

Syntax

```

APEX_ESCAPE.STRIPHTML (
    p_string      IN VARCHAR2 )
    RETURN VARCHAR2 deterministic;

```

Parameters

Parameter	Description
<code>p_string</code>	The input text string.

Example

```
begin
    sys.htp.p(apex_escape.striphtml(
        q'[<p id="greeting">Hello <b>Joe</b></p>]' 
    ));
end;
```

Result:

```
Hello Joe
```

```
begin
    sys.htp.p(apex_escape.striphtml(q' [
        <html>
            <head>
                <title>Web Page</title>
            </head>
            <body>
                <h1>Page Title</h1>
                <p>
                    This is some text.
                </p>
            </body>
        </html>
    ]'));
end;
```

Result:

```
-----
```

```
Web Page
```

```
Page Title
```

```
This is some text.
```

26.27 STRIPHTML Function Signature 2

This function returns `p_string` (CLOB) removing HTML tags, leaving plain text.

This function removes all HTML attributes regardless of the type of HTML content. For example, it preserves content such as JavaScript and CSS, but removes script and CSS HTML tags.

Syntax

```
APEX_ESCAPE.STRIPHTML (
    p_string IN CLOB )
    RETURN CLOB deterministic;
```

Parameters

Parameter	Description
p_string	The input text string.

Example

```
BEGIN
    sys.htp.p(apex_escape.striphtml(
        q'[<p id="greeting">Hello <b>Joe</b></p>]' );
END;
```

Result:

```
-----  
Hello Joe  
-----
```

```
BEGIN
    sys.htp.p(apex_escape.striphtml(q' [
        <html>
            <head>
                <title>Web Page</title>
            </head>
            <body>
                <h1>Page Title</h1>
                <p>
                    This is some text.
                </p>
            </body>
        </html>
    ] '));
END;
```

Result:

```
-----  
Web Page  
  
Page Title  
  
This is some text.
```


APEX_EXEC

The `APEX_EXEC` package encapsulates data processing and querying capabilities and provides an abstraction from the data source to APEX components and plug-ins. `APEX_EXEC` contains procedures and functions to execute queries or procedural calls on local and remote data sources as well as REST Data Sources. It can be used for plug-in development and procedural PL/SQL processing in applications or within packages and procedures.

All `APEX_EXEC` procedures require an existing APEX session to function. In a pure SQL or PL/SQL context, use the `APEX_SESSION` package to initialize a new session.

 **Note:**

Always add an exception handler to your procedure or function to ensure that `APEX_EXEC.CLOSE` is always called to release server resources such as database cursors and temporarylobs.

- Call Sequences for `APEX_EXEC`
- Global Constants
- Data Types
- `ADD_COLUMN` Procedure
- `ADD_DML_ARRAY_ROW` Procedure
- `ADD_DML_ROW` Procedure
- `ADD_FILTER` Procedures
- `ADD_ORDER_BY` Procedure
- `ADD_PARAMETER` Procedure
- `CLEAR_DML_ROWS` Procedure
- `CLOSE` Procedure
- `CLOSE_ARRAY` Procedure
- `COLUMN_EXISTS` Function
- `COPY_DATA` Procedure
- `DESCRIBE_QUERY` Function Signature 1
- `DESCRIBE_QUERY` Function Signature 2
- `ENQUOTE_LITERAL` Function
- `ENQUOTE_NAME` Function
- `EXECUTE_DML` Procedure
- `EXECUTE_PLSQL` Procedure Signature 1
- `EXECUTE_PLSQL` Procedure Signature 2

- EXECUTE_REMOTE_PLSQL Procedure Signature 1
- EXECUTE_REMOTE_PLSQL Procedure Signature 2
- EXECUTE_REST_SOURCE Procedure Signature 1
- EXECUTE_REST_SOURCE Procedure Signature 2
- EXECUTE_WEB_SOURCE Procedure (Deprecated)
- GET Function
- GET_ARRAY_ROW_DML_OPERATION Function
- GET_ARRAY_ROW_VERSION_CHECKSUM Function
- GET_COLUMN Function
- GET_COLUMNS Function
- GET_COLUMN_COUNT Function
- GET_COLUMN_POSITION Function
- GET_DATA_TYPE Function
- GET_DML_STATUS_CODE Function
- GET_DML_STATUS_MESSAGE Function
- GET_PARAMETER Functions
- GET_ROW_VERSION_CHECKSUM Function
- GET_TOTAL_ROW_COUNT Function
- HAS_ERROR Function
- HAS_MORE_ARRAY_ROWS Function
- HAS_MORE_ROWS Function
- IS_GROUP_END Function
- IS_REMOTE_SQL_AUTH_VALID Function
- NEXT_ARRAY_ROW Function
- NEXT_ROW Function
- OPEN_ARRAY Procedure Signature 1
- OPEN_ARRAY Procedure Signature 2
- OPEN_DUALITY_VIEW_DML_CONTEXT Function
- OPEN_JSON_SOURCE_DML_CONTEXT Function
- OPEN_LOCAL_DML_CONTEXT Function
- OPEN_QUERY_CONTEXT Function Signature 1
- OPEN_QUERY_CONTEXT Function Signature 2
- OPEN_REMOTE_DML_CONTEXT Function
- OPEN_REMOTE_SQL_QUERY Function
- OPEN_REST_SOURCE_DML_CONTEXT Function
- OPEN_REST_SOURCE_QUERY Function
- OPEN_WEB_SOURCE_DML_CONTEXT Function (Deprecated)
- OPEN_WEB_SOURCE_QUERY Function (Deprecated)

- PURGE_DUALITY_VIEW_CACHE Procedure
- PURGE_JSON_SOURCE_CACHE Procedure
- PURGE_REST_SOURCE_CACHE Procedure
- PURGE_WEB_SOURCE_CACHE Procedure (Deprecated)
- SET_ARRAY_CURRENT_ROW Procedure
- SET_ARRAY_ROW_VERSION_CHECKSUM Procedure
- SET_CURRENT_ROW Procedure
- SET_NULL Procedure
- SET_ROW_VERSION_CHECKSUM Procedure
- SET_VALUE Procedure
- SET_VALUES Procedure

27.1 Call Sequences for APEX_EXEC

All APEX_EXEC procedures require an existing APEX session to function. In a pure SQL or PL/SQL context, use the APEX_SESSION package to initialize a new session.

- Querying a Data Source with APEX_EXEC
- Executing a DML on a Data Source with APEX_EXEC
- Executing a Remote Procedure or REST API with APEX_EXEC



See Also:

APEX_SESSION

27.1.1 Querying a Data Source with APEX_EXEC

1. Prepare columns to be selected from the data source:
 - a. Create a variable of the APEX_EXEC.T_COLUMNS type.
 - b. Add columns with the APEX_EXEC.ADD_COLUMN.
2. (Optional) Prepare bind variables:
 - a. Create a variable of APEX_EXEC.T_PARAMETERS type.
 - b. Add bind values with APEX_EXEC.ADD_PARAMETER.
3. (Optional) Prepare filters:
 - a. Create a variable of the type APEX_EXEC.T_FILTERS.
 - b. Add bind values with APEX_EXEC.ADD_FILTER.
4. Execute the data source query in one of the following ways:
 - For **REST Data Sources**, use APEX_EXEC.OPEN_REST_SOURCE_QUERY.
 - For **REST Enabled SQL**, use APEX_EXEC.OPEN_REMOTE_SQL_QUERY.
 - Alternatively, use APEX_EXEC.OPEN_QUERY_CONTEXT to pass in the location as a parameter.

5. Get the result set meta data:
 - a. APEX_EXEC.GET_COLUMN_COUNT returns the number of result columns.
 - b. APEX_EXEC.GET_COLUMN returns information about a specific column.
6. Process the result set:
 - a. APEX_EXEC.NEXT_ROW advances the result cursor by one row.
 - b. APEX_EXEC.GET_NNNN functions retrieve individual column values.
7. Close all resources with APEX_EXEC.CLOSE.
8. Add an exception handler and close those resources. For example:

```
EXCEPTION
    WHEN others THEN
        apex_debug.log_exception;
        apex_exec.close( l_context );
    RAISE;
```

See Also:

For code examples of a complete query to a Data Source, review the example sections in the following APIs:

- [OPEN_QUERY_CONTEXT Function Signature 2](#)
- [OPEN_REMOTE_SQL_QUERY Function](#)
- [OPEN_REST_SOURCE_QUERY Function](#)

27.1.2 Executing a DML on a Data Source with APEX_EXEC

1. Define the Data Manipulation Language (DML) columns:
 - a. Create a variable of the APEX_EXEC.T_COLUMNS type.
 - b. Add columns with APEX_EXEC.ADD_COLUMN.
2. (Optional) Prepare bind variables:
 - a. Create a variable of the APEX_EXEC.T_PARAMETERS type.
 - b. Add bind values with APEX_EXEC.ADD_PARAMETER.
3. Prepare the DML Context in one of the following ways:
 - For **REST Data Sources**, use OPEN_REST_SOURCE_DML_CONTEXT.
 - For **REST Enabled SQL**, use OPEN_REMOTE_DML_CONTEXT.
 - For **local database**, use OPEN_LOCAL_DML_CONTEXT.
4. Add row values for the DML to perform:
 - a. Use APEX_EXEC.ADD_DML_ROW to add a new row.
 - b. Use APEX_EXEC.SET_VALUE to provide individual column values.
5. Execute the DML with APEX_EXEC.EXECUTE_DML.

6. Walk through RETURNING values and error messages for processed DML rows.
 - APEX_EXEC.NEXT_ROW advances the result cursor by one row.
 - APEX_EXEC.HAS_ERROR indicates whether DML processing for this row was successful or not.
 - APEX_EXEC.GET_DML_STATUS_CODE returns the status code (SQL Error Code) for each DML row. If DML for this row was successful, NULL is returned as the status code.
 - APEX_EXEC.GET_NNN functions retrieve individual column "DML RETURNING" values.
7. Close all resources with APEX_EXEC.CLOSE.
8. Add an exception handler and close those resources. For example:

```
EXCEPTION
  WHEN others THEN
    apex_exec.close( l_context );
    RAISE;
```

See Also:

For code examples of a complete DML query, review the example sections in the following APIs:

- [OPEN_LOCAL_DML_CONTEXT Function](#)
- [OPEN_REMOTE_DML_CONTEXT Function](#)
- [OPEN_REST_SOURCE_DML_CONTEXT Function](#)

27.1.3 Executing a Remote Procedure or REST API with APEX_EXEC

1. (Optional) Prepare bind variables:
 - a. Create a variable of APEX_EXEC.T_PARAMETERS type.
 - b. Add bind values with APEX_EXEC.ADD_PARAMETER.
2. Execute the local or remote procedure or REST API in one of the following ways:
 - For **REST Data Sources**, use APEX_EXEC.EXECUTE_REST_SOURCE.
 - For **REST Enabled SQL**, use APEX_EXEC.EXECUTE_REMOTE_PLSQL.
 - For **local database**, use APEX_EXEC.EXECUTE_PLSQL.
3. (Optional) Retrieve the OUT parameters. Walk through the variable of the APEX_EXEC.T_PARAMETERS type and use GET_PARAMETER_VALUE to retrieve the OUT parameter value.

The P_PARAMETERS array which is used to pass bind variables is an IN OUT parameter, so OUT parameters are passed back.

 See Also:

For code examples of a complete remote procedure or REST API query, review the example sections in the following APIs:

- [EXECUTE_PLSQL Procedure Signature 1](#)
- [EXECUTE_REMOTE_PLSQL Procedure Signature 1](#)
- [EXECUTE_REST_SOURCE Procedure Signature 1](#)

27.2 Global Constants

The APEX_EXEC package uses the following constants.

Query or DML Location Constants

```
subtype t_location is varchar2(12);

c_location_local_db      constant t_location := 'LOCAL';
c_location_remote_db     constant t_location := 'REMOTE';
c_location_rest_source   constant t_location := 'REST_SOURCE';
c_location_region_source constant t_location := 'REGION_SOURCE';
c_location_duality_view  constant t_location := 'DUALITY_VIEW';
c_location_json_source   constant t_location := 'JSON_COLLECTION';

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_func_return_sql constant t_post_processing := 3;
```

Column Data Type Constants

Data type constants to be used in the ADD_FILTER or ADD_COLUMN procedures.

```
subtype t_data_type is pls_integer range 1..18;

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;
c_data_type_binary_number constant t_data_type := 14;
c_data_type_sdo_geometry   constant t_data_type := 15;
c_data_type_boolean        constant t_data_type := 16; -- only available for
PL/SQL.

c_data_type_array          constant t_data_type := 17;
c_data_type_vector         constant t_data_type := 18;
-- 
-- Use this data type for columns of the JSON data type (Database 21c or
higher) ONLY.
-- Has currently the same functionality as CLOB columns, but may be extended
in the
-- future.
c_data_type_json constant t_data_type := 11;
```

Filter Type Constants

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_between_lbe        constant t_filter_type := 29; -- Interactive report
```

```

filter.
c_filter_between_ube      constant t_filter_type := 30; -- Interactive report
filter.
c_filter_not_between     constant t_filter_type := 18;
c_filter_regexp          constant t_filter_type := 19;
c_filter_last             constant t_filter_type := 20; -- date filter: days/
months/...
c_filter_not_last         constant t_filter_type := 21; -- date filter: days/
months/...
c_filter_next             constant t_filter_type := 22; -- date filter: days/
months/...
c_filter_not_next         constant t_filter_type := 23; -- date filter: days/
months/...
c_filter_like              constant t_filter_type := 24; -- Interactive report
filter.
c_filter_not_like         constant t_filter_type := 25; -- Interactive report
filter.
c_filter_search            constant t_filter_type := 26; -- Interactive report
filter.
c_filter_sql_expression   constant t_filter_type := 27; -- Interactive report
filter.
c_filter_oracletext        constant t_filter_type := 28; -- Oracle TEXT
CONTAINS filter.
c_filter_sdo_filter        constant t_filter_type := 31; -- Spatial filter.
c_filter_sdo_anyinteract   constant t_filter_type := 32; -- Spatial filter.
c_filter_dbms_search       constant t_filter_type := 33; -- Oracle Ubiquitous
Search CONTAINS Filter.
c_filter_vector_type       constant t_filter_type := 34;

```

Order By Constants

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;

```

Order By Nulls Constants

Order By Nulls constants to use within REST Source Plug-Ins.

```

subtype t_supports_orderby_nulls_as is pls_integer range 1..5;

c_orderby_nulls_flexible    constant t_supports_orderby_nulls_as := 1;
c_orderby_nulls_are_lowest   constant t_supports_orderby_nulls_as := 2;
c_orderby_nulls_are_highest  constant t_supports_orderby_nulls_as := 3;
c_orderby_nulls_always_last  constant t_supports_orderby_nulls_as := 4;
c_orderby_nulls_always_first constant t_supports_orderby_nulls_as := 5;

```

Empty Constants

Constants for 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;
```

Database Vendor Constants

```
subtype t_database_type is pls_integer range 1..2;
c_database_oracle constant t_database_type := 1;
c_database_mysql   constant t_database_type := 2;
```

Aggregation Type Constants

```
subtype t_aggregation_type is pls_integer range 1..3;
c_aggregation_none constant    t_aggregation_type := 1;
c_aggregation_group_by constant t_aggregation_type := 2;
c_aggregation_distinct constant t_aggregation_type := 3;
```

Aggregation Column Role Constants

```
subtype t_column_role is pls_integer range 1..2;
c_column_role_aggregate constant t_column_role := 1;
c_column_role_group_by   constant t_column_role := 2;
```

Aggregation Function Constants

```
subtype t_aggregate_function is pls_integer range 1..11;
c_aggregate_sum           constant t_aggregate_function := 1;
c_aggregate_avg            constant t_aggregate_function := 2;
c_aggregate_median          constant t_aggregate_function := 3;
c_aggregate_cnt             constant t_aggregate_function := 4;
c_aggregate_distinct_cnt    constant t_aggregate_function := 5;
c_aggregate_approx_dist_cnt constant t_aggregate_function := 6;
c_aggregate_min              constant t_aggregate_function := 7;
c_aggregate_max              constant t_aggregate_function := 8;
c_aggregate_ratio_report_sum constant t_aggregate_function := 9;
c_aggregate_ratio_report_cnt constant t_aggregate_function := 10;
c_aggregate_listagg         constant t_aggregate_function := 11;
```

Aggregation Columns

```
type t_aggregation_column is record(
    attributes      t_column,
    aggr_role       t_column_role,
    aggr_function   t_aggregate_function,
```

```
total_column_name  t_column_name,
total_function      t_aggregate_function );
```

Collection of Aggregation Columns

```
type t_aggregation_columns is table of t_aggregation_column index by
pls_integer;
```

Aggregation

```
type t_aggregation is record(
    aggregation_type          t_aggregation_type,
    column_info                t_aggregation_columns,
    order_bys                  t_order_bys,
    order_by_expr               varchar2(32767),
    row_count_column            t_column_name );
```



```
c_empty_aggregation t_aggregation;
```

Vector Search Subtypes

```
subtype t_vector_search_type is pls_integer range 1..3;
c_vector_search_exact  constant t_vector_search_type := 1;
c_vector_search_approx constant t_vector_search_type := 2;
c_vector_search_multi   constant t_vector_search_type := 3;

--
subtype t_vector_distance_type is pls_integer range 1..6;
c_vector_distance_cosine      constant t_vector_distance_type := 1;
c_vector_distance_dot         constant t_vector_distance_type := 2;
c_vector_distance_euclidean   constant t_vector_distance_type := 3;
c_vector_distance_eucl_squared constant t_vector_distance_type := 4;
c_vector_distance_hamming     constant t_vector_distance_type := 5;
c_vector_distance_manhattan   constant t_vector_distance_type := 6;
--
subtype t_vector_distance is varchar2(17);

c_vector_dist_str_cosine      constant t_vector_distance := 'COSINE';
c_vector_dist_str_dot         constant t_vector_distance := 'DOT';
c_vector_dist_str_euclidean   constant t_vector_distance := 'EUCLIDEAN';
c_vector_dist_str_eucl_squared constant t_vector_distance := 'EUCLIDEAN_SQUARED';
c_vector_dist_str_hamming     constant t_vector_distance := 'HAMMING';
c_vector_dist_str_manhattan   constant t_vector_distance := 'MANHATTAN';
```

27.3 Data Types

The APEX_EXEC package uses the following data types.

Generic

```
subtype t_column_name is varchar2(32767);

type t_value is record (
    varchar2_value      varchar2(32767),
    number_value        number,
    binary_number_value binary_double,
    date_value          date,
    timestamp_value     timestamp,
    timestamp_tz_value timestamp with time zone,
    timestamp_ltz_value timestamp with local time zone,
    interval_y2m_value yminterval_unconstrained,
    interval_d2s_value dsinterval_unconstrained,
    blob_value          blob,
    bfile_value         bfile,
    clob_value          clob,
    sdo_geometry_value  mdsys.sdo_geometry,
    anydata_value        sys.anydata );

type t_values is table of t_value index by pls_integer;
```

 **Note:**

sdo_geometry_value is **only** available when SDO_GEOmetry is installed in the database.

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

Note:

The data source capabilities filter_* and orderby_* are deprecated and will be removed in a future release.

```
type t_source_capabilities is record (
    location          t_location,
    --
    pagination        boolean default false,
    page_size_is_stable boolean default false,
    --
    allow_fetch_all_rows boolean default false,
```

```

--  

filtering          boolean default false,  

order_by           boolean default false,  

group_by           boolean default false,  

--  

orderby_nulls_as  t_supports_orderby_nulls_as,  

--  

-- the following filter_* attributes are deprecated, do not use.  

--  

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,  

filter_regexp      boolean default false,  

filter_last        boolean default false,  

filter_not_last    boolean default false,  

filter_next        boolean default false,  

filter_not_next    boolean default false,  

--  

-- the following orderby_* attributes are deprecated, do not use.  

--  

orderby_asc        boolean default false,  

orderby_desc       boolean default false,  

orderby_nulls      boolean default false );

```

Column Meta Data

Attribute	Description
name	Column Name or Alias.
parent_column_position	stores the reference to the parent column
data_type	Data Type: Use constants c_data_type_*.
data_type_length	Data Type Length for VARCHAR2 columns.
sql_expression	SQL Expression for derived columns.
format_mask	Format Mask for NUMBER, DATE or TIMESTAMP columns.
is_required	Whether the column is required (NOT NULL)
is_primary_key	Whether the column is part of the table primary key
is_query_only	Query Only columns are not part of DML operations.

Attribute	Description
is_checksum	Whether the column is designated as the Row Version column.
is_returning	Whether the new value is to be returned after a DML operation.

```
type t_column is record (
    name                  t_column_name,
    --
    parent_column_position  pls_integer,
    --
    data_type              t_data_type,
    data_type_length        pls_integer,
    --
    sql_expression          varchar2(32767),
    --
    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 );
```

27.4 ADD_COLUMN Procedure

This procedure adds a column to the columns collection.

Columns collections can be passed to the OPEN_*_CONTEXT calls in order to request only a subset of columns. This is particularly useful for REST Data Sources without a SQL statement. If no or an empty column array is passed, all columns defined in the web source are fetched.

Syntax

```
APEX_EXEC.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,
    p_parent_column_path IN          VARCHAR2    DEFAULT NULL );
```

Parameters

Parameter	Description
p_columns	Columns array.
p_column_name	Column name.

Parameter	Description
p_data_type	Column data type.
p_sql_expression	SQL expression used to derive a column from other columns.
p_format_mask	Format mask to use for this column.
p_is_primary_key	Whether this is a primary key column (default FALSE).
p_is_query_only	Query only columns are not written in a DML context (default FALSE).
p_is_returning	Whether to retrieve the RETURNING column after DML has been executed (default FALSE).
p_is_checksum	Whether this is a checksum (row version) column (default FALSE).
p_parent_column_path	Path to the parent column to look the index up within.

Example

```

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_rest_source_query(
        p_module_static_id => '{REST Data Source 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;

```

27.5 ADD_DML_ARRAY_ROW Procedure

This procedure adds a child row for the current array or the array column provided as p_column_name. The cursor moves to the new row within the specified array column, and all subsequent calls to SET_VALUE target the attributes of this new array element. Only supported within DML contexts on REST Data Sources.

Hierarchical structures are currently only supported for DML on REST Data Sources, if the REST Source type or Plug-In can deal with such structures. DML on a local table or based on REST-Enabled SQL ignores array columns.

The provided array column must be a direct child of the current array column; path syntax and jumping to another position in the hierarchy is unsupported.

Syntax

```
APEX_EXEC.ADD_DML_ARRAY_ROW (
    p_context          IN t_context,
    p_column_name      IN VARCHAR2        DEFAULT NULL,
    p_column_position  IN PLS_INTEGER,
    p_operation        IN t_dml_operation DEFAULT NULL )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_column_name	Name of the array column (must exist within the current context) to add a new row for.
p_column_position	Position of the column to set the value for within the DML context.
p_operation	DML operation to be executed on this row. Use constants c_dml_operation_*. If omitted, the child row inherits the operation from its parent.

Example

```
declare
    l_columns apex_exec.t_columns;
    l_context apex_exec.t_context;

begin

    --
    -- I. Define DML columns
    --
    -- 1. row-level columns
    --
    apex_exec.add_column(
        p_columns      => l_columns,
        p_column_name  => 'CUSTOMER_NAME',
        p_data_type    => apex_exec.c_data_type_varchar2 );

    apex_exec.add_column(
        p_columns      => l_columns,
        p_column_name  => 'ORDER_DATE',
        p_data_type    => apex_exec.c_data_type_date );

    apex_exec.add_column(
        p_columns      => l_columns,
        p_column_name  => 'ORDER_ITEMS',
        p_data_type    => apex_exec.c_data_type_array );
```

```

--  

-- 2. child columns of the ORDER_ITEMS array column  

--  

apex_exec.add_column(  

    p_columns          => l_columns,  

    p_column_name     => 'PRODUCT_ID',  

    p_data_type       => apex_exec.c_data_type_number,  

    p_parent_column_path => 'ORDER_ITEMS' );  

  

apex_exec.add_column(  

    p_columns          => l_columns,  

    p_column_name     => 'PRODUCT_NAME',  

    p_data_type       => apex_exec.c_data_type_varchar2,  

    p_parent_column_path => 'ORDER_ITEMS' );  

  

apex_exec.add_column(  

    p_columns          => l_columns,  

    p_column_name     => 'UNIT_PRICE',  

    p_data_type       => apex_exec.c_data_type_number,  

    p_parent_column_path => 'ORDER_ITEMS' );  

  

apex_exec.add_column(  

    p_columns          => l_columns,  

    p_column_name     => 'AMOUNT_ORDERED',  

    p_data_type       => apex_exec.c_data_type_number,  

    p_parent_column_path => 'ORDER_ITEMS' );  

  

--  

-- II. Open the context object  

--  

l_context := apex_exec.open_rest_source_dml_context(  

    p_columns      => l_columns,  

    p_static_id   => '{module static id}' );  

  

--  

-- III: Provide DML data  

--  

-- 1. the first row  

--  

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_name  => 'CUSTOMER_NAME',  

    p_value        => 'John Doe' );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'ORDER_DATE',  

    p_value        => date'2024-03-15' );  

  

--  

-- 1.1. the first line item of the first row

```

```

--  

apex_exec.add_dml_array_row(  

    p_context      => l_context,  

    p_operation    => apex_exec.c_dml_operation_insert,  

    p_column_name  => 'ORDER_ITEMS');  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'PRODUCT_ID',  

    p_value        => 100 );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'PRODUCT_NAME',  

    p_value        => 'Men''s Jeans size L' );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'UNIT_PRICE',  

    p_value        => 30.99 );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'AMOUNT_ORDERED',  

    p_value        => 10 );  

  

--  

-- 1.2. the second line item of the first row  

--  

apex_exec.add_dml_array_row(  

    p_context      => l_context );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'PRODUCT_ID',  

    p_value        => 101 );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'PRODUCT_NAME',  

    p_value        => 'Ladies Jeans size S' );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'UNIT_PRICE',  

    p_value        => 30.99 );  

  

apex_exec.set_value(  

    p_context      => l_context,  

    p_column_name  => 'AMOUNT_ORDERED',  

    p_value        => 10 );  

  

--  

-- 2. the second row  

--  

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_name  => 'CUSTOMER_NAME',
    p_value        => 'Jane Doe' );

apex_exec.set_value(
    p_context      => l_context,
    p_column_name  => 'ORDER_DATE',
    p_value        => date'2024-03-16' );

-- 
-- 2.1. the first line item of the second row
--

apex_exec.add_dml_array_row(
    p_context      => l_context,
    p_operation    => apex_exec.c_dml_operation_insert,
    p_column_name  => 'ORDER_ITEMS');

apex_exec.set_value(
    p_context      => l_context,
    p_column_name  => 'PRODUCT_ID',
    p_value        => 100 );

--- :

apex_exec.add_dml_array_row(
    p_context      => l_context,
    p_operation    => apex_exec.c_dml_operation_insert );

--- :

-- IV: Set "cursor" back to the first child in order to change a value

apex_exec.set_array_current_row(
    p_context      => l_context,
    p_current_row_idx => 1 );

apex_exec.set_value(
    p_context      => l_context,
    p_column_name  => 'AMOUNT_ORDERED',
    p_value        => 20 );

-- V: 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;
```

 **See Also:**

- [CLOSE_ARRAY Procedure](#)
- [OPEN_ARRAY Procedure Signature 1](#)
- [NEXT_ARRAY_ROW Function](#)
- [SET_ARRAY_CURRENT_ROW Procedure](#)
- [GET_ARRAY_ROW_DML_OPERATION Function](#)

27.6 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

```
APEX_EXEC.ADD_DML_ROW (  
    p_context          IN t_context,  
    p_operation        IN t_dml_operation );
```

Parameters

Parameter	Description
<code>p_context</code>	Context object obtained with one of the <code>OPEN_</code> functions
<code>p_operation</code>	DML operation to be executed on this row. Possible values: <ul style="list-style-type: none">• <code>c_dml_operation_insert</code>• <code>c_dml_operation_update</code>• <code>c_dml_operation_delete</code>

 **See Also:**

- [OPEN_REMOTE_DML_CONTEXT Function](#)
- [OPEN_WEB_SOURCE_DML_CONTEXT Function \(Deprecated\)](#)
- [OPEN_LOCAL_DML_CONTEXT Function](#)

27.7 ADD_FILTER Procedures

This procedure adds a filter to the filter collection.

Syntax

Signature 1

```
PROCEDURE ADD_FILTER (
    p_filters          IN OUT NOCOPY t_filters,
    p_filter_type     IN          t_filter_type,
    p_column_name     IN          t_column_name );
```

Signature 2

```
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          apex_t_varchar2,
    p_null_result    IN          BOOLEAN DEFAULT FALSE,
    p_is_case_sensitive IN        BOOLEAN DEFAULT TRUE,
    p_data_type       IN          t_data_type DEFAULT
c_data_type_varchar2 );
```

Signature 3

```
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 );
```

Signature 4

```
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          apex_t_varchar2,
    p_null_result    IN          BOOLEAN DEFAULT FALSE,
    p_is_case_sensitive IN        BOOLEAN DEFAULT TRUE );
```

Signature 5

```
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 );

```

Signature 6

```

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 );

```

Signature 7

```

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      apex_t_number,
    p_null_result   IN      BOOLEAN DEFAULT FALSE );

```

Signature 8

```

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 );

```

Signature 9

```

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 );

```

Signature 10

```

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 );

```

Signature 11

```
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 );
```

Signature 12

```
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 WITH TIME ZONE,
    p_null_result     IN          BOOLEAN DEFAULT FALSE );
```

Signature 13

```
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 );
```

Signature 14

```
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 WITH LOCAL TIME ZONE,
    p_null_result     IN          BOOLEAN DEFAULT FALSE );
```

Signature 15

```
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 );
```

Signature 16

```
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,
p_data_type        IN          t_data_type  DEFAULT c_data_type_date );

```

Signature 17

```

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,
    p_tokenize         IN          BOOLEAN      DEFAULT NULL );

```

Signature 18

```

PROCEDURE ADD_FILTER (
    p_filters          IN OUT NOCOPY t_filters,
    p_sql_expression   IN          VARCHAR2 );

```

Signature 19

Note:

This signature is **only** available if SDO_GEOmetry (Oracle Locator) is installed in the database.

```

PROCEDURE ADD_FILTER (
    p_filters          IN OUT NOCOPY t_filters,
    p_filter_type     IN          t_filter_type,
    p_column_name     IN          VARCHAR2,
    p_value           IN          mdsys.sdo_geometry );

```

Signature 20

```

PROCEDURE ADD_FILTER (
    p_filters          IN OUT NOCOPY t_filters,
    p_search_index_owner IN          VARCHAR2,
    p_search_index_table IN          VARCHAR2,
    p_text_column_name  IN          VARCHAR2,
    p_text_query_function IN          VARCHAR2,
    p_value            IN          VARCHAR2 );

```

Signature 21

 **Note:**

Use this signature for Oracle TEXT.

```
PROCEDURE ADD_FILTER (
    p_filters          IN OUT NOCOPY t_filters,
    p_text_column_name IN              VARCHAR2,
    p_text_query_function IN          VARCHAR2,
    p_value             IN              VARCHAR2 );
```

Signature 22

 **Note:**

Requires Database 23ai and specific add_column procedure:

```
apex_exec.add_vector_distance_column(
    p_columns           => l_columns );
```

```
PROCEDURE ADD_FILTER (
    p_filters          IN OUT NOCOPY t_filters,
    p_vector_column_name IN              VARCHAR2,
    --
    p_vector_search_type IN              t_vector_search_type      DEFAULT
c_vector_search_exact,
    p_distance_metric     IN              t_vector_distance_type DEFAULT
c_vector_distance_euclidean,
    --
    -- only for p_vector_search_type = c_vector_search_approx
    p_target_accuracy    IN              NUMBER                  DEFAULT NULL,
    --
    p_max_results        IN              NUMBER,
    p_max_dist           IN              NUMBER                  DEFAULT NULL,
    p_search_vector       IN              VECTOR ) ;
```

Parameters

Parameter	Description
p_filters	Filters array.
p_filter_type	Type of filter - use one of the t_filter_type constants.
p_column_name	Column to apply this filter on.
p_value	Value for filters requiring one value (for example, equals or greater than).
p_values	Value array for IN or NOT IN filters.
p_from_value	Lower value for filters requiring a range (for example, between).
p_to_value	Upper value for filters requiring a range (for example, between).

Parameter	Description
p_interval	Interval for date filters (for example, last X months).
p_interval_type	Interval type for date filters (months, dates).
p_sql_expression	Generic SQL expression to use as filter.
p_null_result	Result to return when the actual column value is NULL.
p_is_case_sensitive	Whether this filter should work case-sensitive or not.
p_search_columns	List of columns to apply the row search filter on.
p_text_column_name	Column name for the SQL contains expression when using Oracle TEXT or Ubiquitous Database Search.
p_text_query_function	Function to be used for the SQL contains expression when using Oracle TEXT or Ubiquitous Database Search.
p_search_index_owner	For Ubiquitous Database Search, to apply a filter for the Ubiquitous Search index source owner.
p_search_index_table	For Ubiquitous Database Search, to apply a filter for the Ubiquitous Search index source name.
p_vector_column_name	Vector column to apply this filter on.
p_vector_search_type	Search Type. Use one of the t_vector_search_type constants.
p_distance_metric	Distance Metric. Use one of the t_vector_distance_type constants.
p_target_accuracy	Target accuracy. Only used if p_vector_search_type = c_vector_search_approx.
p_max_results	Amount of rows to fetch.
p_max_dist	Maximum Vector Distance for the search results.
p_search_vector	Vector value for the Vector Search.
p_data_type	Data type of the column to apply this filter on.
p_tokenize	Whether to tokenize a row search term to individual words.

Example

```

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 );
        raise;
END;

```

27.8 ADD_ORDER_BY Procedure

This procedure adds an order by expression to the order bys collection.

Syntax

Signature 1

```
APEX_EXEC.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 )
```

Signature 2

```
APEX_EXEC.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

Parameter	Description
p_order_bys	Order by collection.
p_position	References a column of the provided data source by position.
p_column_name	References a column name or alias of the provided data source.
p_direction	Defines if the column is sorted ascending or descending. Valid values are c_order_asc and c_order_desc.
p_order_nulls	Defines if NULL data sorts 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.

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;

```

27.9 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

Signature 1

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          VARCHAR2 )
```

Signature 2

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          NUMBER )
```

Signature 3

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          DATE )
```

Signature 4

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
```

```

    p_name      IN          t_column_name,
    p_value     IN          TIMESTAMP )

```

Signature 5

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          TIMESTAMP WITH TIME ZONE )

```

Signature 6

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       in          t_column_name,
    p_value      IN          TIMESTAMP WITH LOCAL TIME ZONE )

```

Signature 7

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       in          t_column_name,
    p_value      in          INTERVAL YEAR TO MONTH )

```

Signature 8

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       in          t_column_name,
    p_value      in          INTERVAL DAY TO SECOND )

```

Signature 9

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          BLOB )

```

Signature 10

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          bfile )

```

Signature 11

```

APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name       IN          t_column_name,
    p_value      IN          CLOB )

```

Signature 12

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name        IN          t_column_name,
    p_value       IN          sys.anydata )
```

Signature 13

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name        IN          t_column_name,
    p_data_type   IN          t_data_type,
    p_value       IN          t_value )
```

Signature 14 **Note:**

This signature is **only** available if SDO_GEOmetry (Oracle Locator) is installed in the database.

```
APEX_EXEC.ADD_PARAMETER (
    p_parameters IN OUT NOCOPY t_parameters,
    p_name        IN          t_column_name,
    p_value       IN          mdsys.sdo_geometry )
```

Parameters

Parameter	Description
p_parameters	SQL parameter array.
p_name	Parameter name.
p_value	Parameter value.

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;
```

27.10 CLEAR_DML_ROWS Procedure

This procedure clears all DML rows which have been added with `add_dml_rows`.

Syntax

```
APEX_EXEC.CLEAR_DML_ROWS (
    p_context           IN t_context )
```

Parameters

Parameter	Description
<code>p_context</code>	Context object obtained with one of the <code>OPEN_</code> functions

27.11 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

```
APEX_EXEC.CLOSE (
    p_context IN t_context )
```

Parameters

Parameter	Description
<code>p_context</code>	Context object obtained with one of the <code>OPEN_</code> functions.

27.12 CLOSE_ARRAY Procedure

This procedure closes the current array and returns the cursor back to the parent element. Subsequent calls to `SET_VALUE` target the attributes of the parent element or root row.

Can only be called after calling `add_dml_array_row` or `open_array`.

An error is raised if called when the cursor is on the root level of the row.

Currently only supported for contexts on REST data sources.

Syntax

```
APEX_EXEC.CLOSE_ARRAY (
    p_context           IN t_context )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

 **See Also:**

- [OPEN_ARRAY Procedure Signature 1](#)
- [NEXT_ARRAY_ROW Function](#)
- [SET_ARRAY_CURRENT_ROW Procedure](#)
- [ADD_DML_ARRAY_ROW Procedure](#)

27.13 COLUMN_EXISTS Function

This function checks whether a column already exists in the columns array.

Syntax

```
APEX_EXEC.COLUMN_EXISTS (
    p_columns          IN t_columns,
    p_column_name      IN VARCHAR2,
    p_parent_column_path IN VARCHAR2 DEFAULT NULL )
    RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_columns	Columns array.
p_column_name	Column name.
p_parent_column_path	Path to the parent column to look the index up within.

Returns

TRUE if the column exists, FALSE otherwise.

Example

The following example builds a column array and verifies that the SAL column exists in the array.

```
DECLARE
    l_columns      apex_exec.t_columns;
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' );
IF apex_exec.column_exists(
    p_columns      => l_columns,
    p_column_name => 'SAL' )
THEN
    -- the column exists ...
END IF;
END;

```

27.14 COPY_DATA Procedure

This procedure fetches all rows from the source context and writes to the target context. Useful for copying data between different data sources (such as local to remote, remote to web source).

Array columns are not supported by the COPY_DATA procedure at this time. In the future, these will be handled as CLOBs in JSON format.

Syntax

```
APEX_EXEC.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

Parameter	Description
p_from_context	Query context to fetch rows from.
p_to_context	DML context to write rows to.
p_operation_column_name	Column in the query context to indicate the DML operation to execute on the target context. Possible values are: <ul style="list-style-type: none"> • "I": insert the row on the target (DML) context • "U": update the row on the target (DML) context • "D": delete the row on the target (DML) context

Example

```

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;

```

27.15 DESCRIBE_QUERY Function Signature 1

This procedure describes the query context based on the current region source.

Syntax

```
APEX_EXEC.DESCRIBE_QUERY (
    p_columns           IN t_columns      DEFAULT c_empty_columns )
    RETURN t_columns;
```

Parameters

Parameter	Description
p_columns	Columns to be selected from the data source.

Returns

The `t_columns` object describing the columns and data types.

Example

The following example describes a query and prints out result column names.

```
DECLARE
    l_columns apex_exec.t_columns;
BEGIN
    l_columns := apex_exec.describe_query;

    FOR i in 1 .. l_columns.count LOOP
        http.p( 'Column #' || i || ': ' ||
apex_escape.html( l_columns( i ).name ) );
    END LOOP;
END;
```

27.16 DESCRIBE_QUERY Function Signature 2

This procedure describes the query context based on the current region source.

Syntax

```
APEX_EXEC.DESCRIBE_QUERY (
    p_location          IN t_location,
    --
    p_table_owner       IN VARCHAR2      DEFAULT NULL,
    p_table_name        IN VARCHAR2      DEFAULT NULL,
    p_match_clause      IN VARCHAR2      DEFAULT NULL,
    p_columns_clause    IN VARCHAR2      DEFAULT NULL,
    p_test_for_rowid   IN BOOLEAN       DEFAULT FALSE,
    --
    p_sql_query         IN VARCHAR2      DEFAULT NULL,
    p_function_body     IN VARCHAR2      DEFAULT NULL,
    p_function_body_language IN t_language DEFAULT c_lang_plsql,
```

```

--          p_optimizer_hint      IN VARCHAR2      DEFAULT NULL,
--          p_server_static_id   IN VARCHAR2      DEFAULT NULL,
--          p_module_static_id   IN VARCHAR2      DEFAULT NULL,
p_post_process_type    IN t_post_processing  DEFAULT NULL,
--          p_columns            IN t_columns       DEFAULT c_empty_columns,
--          p_duality_view_static_id IN VARCHAR2      DEFAULT NULL,
p_json_source_static_id IN VARCHAR2      DEFAULT NULL )
RETURN t_columns;

```

Parameters

Parameter	Description
p_location	Location to open the query context for. Use constants c_location_*.
p_table_owner	Table owner when query type TABLE is used.
p_table_name	Table name when query type TABLE is used.
p_match_clause	Match clause to append when query type GRAPH is used.
p_columns_clause	Columns clause to append when query type GRAPH is used.
p_test_for_rowid	Whether to attempt including the ROWID column to the query being described.
p_sql_query	SQL Query to execute when query type SQL Query is used.
p_function_body	Function body returning SQL query.
p_function_body_language	Programming Language used for p_function_body. Use constants c_lang_*
p_optimizer_hint	Optimizer hint to be applied to the most outer SQL query generated by Oracle APEX.
p_server_static_id	Static ID of the Remote Server when REST-Enabled SQL is used.
p_module_static_id	Static ID of the REST data source.
p_post_process_type	Type of post processing to be applied to the REST data source result data.
p_columns	Columns to be selected from the data source.
p_duality_view_static_id	Static ID of the Duality View source.
p_json_source_static_id	Static ID of the JSON source.

Returns

The `t_columns` object describing the columns and data types.

Example

The following example describes a query and prints out result column names.

```

DECLARE
    l_columns apex_exec.t_columns;
BEGIN
    l_columns := apex_exec.describe_query(

```

```

    p_location => apex_exec.c_location_local_db,
    p_sql_query => 'select * from emp' );

    FOR i in 1 .. l_columns.count LOOP
        htp.p( 'Col #' || i || ':' || apex_escape.html( l_columns( i ).name ) );
    END LOOP;
END;

```

27.17 ENQUOTE_LITERAL Function

This function enquotes a string literal and escape contained quotes. This function works for all database types supported by Oracle APEX over REST-enabled SQL.

Syntax

```
APEX_EXEC.ENQUOTE_LITERAL (
    p_str          IN VARCHAR2,
    p_for_database IN t_database_type DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_str	String literal to enquote.
p_for_database	Target database to enquote for.
	If omitted, the function enquotes for the target database of the currently executed region.

Returns

This function returns the enquoted string literal.

Example

```

DECLARE
    l_enquoted_literal varchar2(32767);
BEGIN
    l_enquoted_literal := apex_exec.enquote_literal(
        p_str          => q'#O'Neil \n#',
        p_for_database => c_database_oracle );

    -- returns: 'O''Neil \n'

    l_enquoted_literal := apex_exec.enquote_literal(
        p_str          => q'#O'Neil \n#',
        p_for_database => c_database_mysql );

    -- returns: 'O''Neil \\n'
END;

```

27.18 ENQUOTE_NAME Function

This function enquotes a database object name and (if applicable) escape contained quotes. This function works for all database types supported by Oracle APEX over REST-enabled SQL.

Syntax

```
APEX_EXEC.ENQUOTE_NAME (
    p_str          IN VARCHAR2,
    p_for_database IN t_database_type DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_str	Object name to enquote.
p_for_database	Target database to enquote for.
	If omitted, the function enquotes for the target database of the currently executed region.

Returns

This function returns the enquoted object name.

Example

```
DECLARE
    l_enquoted_literal varchar2(32767);
BEGIN
    l_enquoted_literal := apex_exec.enquote_name(
        p_str          => q'emp',
        p_for_database => c_database_oracle );

    -- returns: "emp"

    l_enquoted_literal := apex_exec.enquote_name(
        p_str          => q'emp#',
        p_for_database => c_database_mysql );

    -- returns: `emp`
END;
```

27.19 EXECUTE_DML Procedure

This procedure executes the DML context. This procedure is called after:

- the context has been opened (`open_dml_context`)
- one or more DML rows have been added with `add_dml_row`
- column values have been set with `set_values`, `set_null` or `set_value`

Syntax

```
APEX_EXEC.EXECUTE_DML (
    p_context          IN t_context,
    p_continue_on_error IN BOOLEAN DEFAULT FALSE )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_continue_on_error	Whether to continue executing DML for the remaining rows after an error occurred. Default FALSE.

Example

An example of this procedure can be found in the examples for the following APIs:

- [SET_ROW_VERSION_CHECKSUM Procedure](#)
- [OPEN_WEB_SOURCE_DML_CONTEXT Function \(Deprecated\)](#)
- [OPEN_LOCAL_DML_CONTEXT Function](#)
- [OPEN_REMOTE_DML_CONTEXT Function](#)

27.20 EXECUTE_PLSQL Procedure Signature 1

This procedure executes PL/SQL code based on the current process or plug-in location settings.

Syntax

```
APEX_EXEC.EXECUTE_PLSQL (
    p_plsql_code      IN      VARCHAR2,
    p_auto_bind_items IN      BOOLEAN      DEFAULT TRUE,
    p_sql_parameters  IN OUT t_parameters )
```

Parameters

Parameter	Description
p_plsql_code	PL/SQL code to execute. Based on the settings of the current process or process-type plug-in, the code is executed locally or remote.
p_auto_bind_items	Default TRUE. 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.
p_sql_parameters	Additional bind variables, if needed. Note that EXECUTE_PLSQL binds all p_sql_parameters as VARCHAR2. Bind variables such as NUMBER and DATE are implicitly converted to VARCHAR2.

Example

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;

```

27.21 EXECUTE_PLSQL Procedure Signature 2

This procedure executes PL/SQL code based on the current process or plug-in location settings.

Syntax

```
APEX_EXEC.EXECUTE_PLSQL (
    p_plsql_code      IN      VARCHAR2 )
```

Parameters

Parameter	Description
p_plsql_code	PL/SQL code to execute. Based on the settings of the current process or process-type plug-in, the code is executed locally or remote.

Example

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;

```

27.22 EXECUTE_REMOTE_PLSQL Procedure Signature 1

This procedure executes PL/SQL code on a REST-enabled SQL instance.

Syntax

```

APEX_EXEC.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

Parameter	Description
p_server_static_id	Static ID of the ORDS REST-enabled SQL instance.
p_plsql_code	PL/SQL code to execute.
p_auto_bind_items	Default TRUE. Whether to automatically bind page item values for both IN and OUT direction. If the PL/SQL code references bind variables which are not page items, this must be set to FALSE.
p_sql_parameters	(Optional) Additional bind variables.

Example

Executes a PL/SQL block on a remote database. 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;
```

27.23 EXECUTE_REMOTE_PLSQL Procedure Signature 2

This procedure executes PL/SQL code on a REST-enabled SQL instance.

Syntax

```
APEX_EXEC.EXECUTE_REMOTE_PLSQL (
    p_server_static_id      IN      VARCHAR2,
    p_plsql_code            IN      VARCHAR2 )
```

Parameters

Parameter	Description
p_server_static_id	Static ID of the ORDS REST-enabled SQL instance.
p_plsql_code	PL/SQL code to execute.

Example

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;
```

27.24 EXECUTE_REST_SOURCE Procedure Signature 1

This procedure executes a REST Source operation based on module static ID, operation, and URL pattern (if required). Use the `t_parameters` array to pass in values for declared REST Data Source parameters. REST Source invocation is based on metadata defined in Shared Components.

Syntax

```
APEX_EXEC.EXECUTE_REST_SOURCE (
    p_static_id      IN      VARCHAR2,
    p_operation      IN      VARCHAR2,
    p_url_pattern   IN      VARCHAR2 DEFAULT NULL,
    p_parameters     IN OUT t_parameters );
```

Parameters

Parameter	Description
p_static_id	Static ID of the REST Data Source.
p_operation	Name of the operation (for example, POST, GET, DELETE).

Parameter	Description
p_url_pattern	If multiple operations with the same name exist, specify the URL pattern, as defined in Shared Components, to identify the REST Source operation.
p_parameters	Parameter values to pass to the external REST Data Source. Note that HTTP Headers, URL Patterns and other parameters being passed to a REST Data Source are typically strings. Oracle recommends that you explicitly pass all values to VARCHAR2 before adding to the t_parameters array.
t_parameters	Array with OUT parameter values, received from the REST Data Source.

Returns

Return	Description
p_parameters	Array with OUT parameter values, received from the REST Data Source.

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 REST Data Source 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:

Name	Direction	Type	Default Value
EMPNO	IN	Request Body	n/a
ENAME	IN	Request Body	n/a
SAL	IN	Request Body	n/a
JOB	IN	Request Body	n/a
RESPONSE	OUT	Request Body	n/a
Content-Type	IN	HTTP Header	application/json

PL/SQL code to invoke that REST 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_rest_source(
        p_static_id      => 'ORDS_EMP',
        p_operation       => 'POST',
```

```

    p_parameters      => l_params );
END;
:P2_RESPONSE := apex_exec.get_parameter_clob(l_params, 'RESPONSE');

```

27.25 EXECUTE_REST_SOURCE Procedure Signature 2

This procedure executes a REST Source operation. The REST Source module and operation are identified by its static IDs. Use the `t_parameters` array to pass in values for declared REST Data Source parameters. REST Source invocation is based on metadata defined in Shared Components.

Syntax

```
APEX_EXEC.EXECUTE_REST_SOURCE (
    p_static_id          IN          VARCHAR2,
    p_operation_static_id IN          VARCHAR2,
    p_parameters         IN OUT NOCOPY t_parameters );
```

Parameters

Parameter	Description
<code>p_static_id</code>	Static ID of the REST Data Source.
<code>p_operation_static_id</code>	Static ID of the operation within the REST Data Source.
<code>p_parameters</code>	Parameter values to pass to the external REST Data Source. Note that HTTP Headers, URL Patterns and other parameters being passed to a REST Data Source are typically strings. Oracle recommends that you explicitly pass all values to <code>VARCHAR2</code> before adding to the <code>t_parameters</code> array.

27.26 EXECUTE_WEB_SOURCE Procedure (Deprecated)

Note:

This procedure is deprecated and will be removed in a future release. Use `execute_rest_source` instead.

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

```
APEX_EXEC.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

Parameter	Description
p_module_static_id	Static ID of the web source module.
p_operation	Name of the operation (for example, POST, GET, DELETE).
p_url_pattern	If multiple operations with the same name exist, specify the URL pattern, as defined in Shared Components, to identify the web source operation.
p_parameters	Parameter values to pass to the external web source. 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.

Returns

Return	Description
p_parameters	Array with OUT parameter values, received from the web source module.

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:

Name	Direction	Type	Default Value
EMPNO	IN	Request Body	n/a
ENAME	IN	Request Body	n/a
SAL	IN	Request Body	n/a
JOB	IN	Request Body	n/a
RESPONSE	OUT	Request Body	n/a
Content-Type	IN	HTTP Header	application/json

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;
```

27.27 GET Function

This function retrieves column values for different data types.

Syntax

Signature 1

```
APEX_EXEC.GET_VARCHAR2 (
    p_context      IN t_context,
    p_column_idx   IN PLS_INTEGER ) RETURN VARCHAR2;

APEX_EXEC.GET_VARCHAR2 (
    p_context      IN t_context,
    p_column_name  IN VARCHAR2 ) RETURN VARCHAR2;
```

Signature 2

```
APEX_EXEC.GET_NUMBER (
    p_context      IN t_context,
    p_column_idx   IN PLS_INTEGER ) RETURN NUMBER;

APEX_EXEC.GET_NUMBER (
    p_context      IN t_context,
    p_column_name  IN VARCHAR2 ) RETURN NUMBER;
```

Signature 3

```
APEX_EXEC.GET_DATE (
    p_context      IN t_context,
    p_column_idx   IN PLS_INTEGER ) RETURN DATE;

APEX_EXEC.GET_DATE (
    p_context      IN t_context,
    p_column_name  IN VARCHAR2 ) RETURN DATE;
```

Signature 4

```
APEX_EXEC.GET_TIMESTAMP (
    p_context      IN t_context,
    p_column_idx   IN PLS_INTEGER ) RETURN TIMESTAMP;

APEX_EXEC.GET_TIMESTAMP (
```

```
p_context      IN t_context,
p_column_name IN VARCHAR2 ) RETURN TIMESTAMP;
```

Signature 5

```
APEX_EXEC.GET_TIMESTAMP_TZ (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN TIMESTAMP WITH TIME ZONE;

APEX_EXEC.GET_TIMESTAMP_TZ (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN TIMESTAMP WITH TIME ZONE;
```

Signature 6

```
APEX_EXEC.GET_TIMESTAMP_LTZ (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN TIMESTAMP WITH LOCAL TIME ZONE;

APEX_EXEC.GET_TIMESTAMP_LTZ (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN TIMESTAMP WITH LOCAL TIME ZONE;
```

Signature 7

```
APEX_EXEC.GET_CLOB (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN CLOB;

APEX_EXEC.GET_CLOB (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN CLOB;
```

Signature 8

```
APEX_EXEC.GET_BLOB (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN BLOB;

APEX_EXEC.GET_BLOB (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN BLOB;
```

Signature 9

```
APEX_EXEC.GET_INTERVAL2S (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN DSINTERVAL_UNCONSTRAINED;

APEX_EXEC.GET_INTERVAL2S (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN DSINTERVAL_UNCONSTRAINED;
```

Signature 10

```
APEX_EXEC.GET_INTERVALY2M (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN YMINTERVAL_UNCONSTRAINED;

APEX_EXEC.GET_INTERVALY2M (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN YMINTERVAL_UNCONSTRAINED;
```

Signature 11

```
APEX_EXEC.GET_ANYDATA (
    p_context      IN t_context,
    p_column_idx  IN PLS_INTEGER ) RETURN SYS.ANYDATA;

APEX_EXEC.GET_ANYDATA (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN SYS.ANYDATA;
```

Signature 12

```
APEX_EXEC.GET_SDO_GEOMETRY (
    p_context      IN t_context,
    p_column_name IN VARCHAR2 ) RETURN MDSYS.SDO_GEOMETRY;
```

 **Note:**

This signature is **only** available if SDO_Geometry (Oracle Locator) is installed in the database.

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_column_idx	Column index.
p_column_name	Column name.

Returns

The column value as specific data type.

27.28 GET_ARRAY_ROW_DML_OPERATION Function

This function returns the DML operation type for the current array element. Can only be called when being inside an array column; otherwise an error message is called.

To be used within a REST Data Source Plug-In when plug-in actions are to be executed based on the DML type for an array element.

Syntax

```
APEX_EXEC.GET_ARRAY_ROW_DML_OPERATION (
    p_context           IN t_context )
    RETURN t_dml_operation;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

The DML type for the current array row as an instance of t_dml_operation.

See Also:

- [ADD_DML_ARRAY_ROW Procedure](#)

27.29 GET_ARRAY_ROW_VERSION_CHECKSUM Function

This function returns the row version checksum for the current nested array row. Can only be called when inside an array column; otherwise an error message is called.

To be used within a REST Data Source Plug-In when a checksum for an array element is needed to perform plug-in actions.

Syntax

```
APEX_EXEC.GET_ARRAY_ROW_VERSION_CHECKSUM (
    p_context           IN t_context )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions

Returns

Row version checksum for the nested current array row.

See Also:

- [SET_ARRAY_ROW_VERSION_CHECKSUM Procedure](#)

27.30 GET_COLUMN Function

This function returns detailed information about a result set column.

Syntax

```
APEX_EXEC.GET_COLUMN (
    p_context      IN t_context,
    p_column_idx   IN PLS_INTEGER )
RETURN t_column;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_column_idx	Index of the column to retrieve information for.

Returns

t_column object with column metadata.

27.31 GET_COLUMNS Function

This function returns the list of columns containing detailed information about each column.

Syntax

```
APEX_EXEC.GET_COLUMNS (
    p_context      IN t_context )
RETURN t_columns;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

t_columns object with column meta data.

27.32 GET_COLUMN_COUNT Function

This function returns the result column count for the current execution context.

Syntax

```
APEX_EXEC.GET_COLUMN_COUNT (
    p_context      IN t_context )
RETURN PLS_INTEGER;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

Returns the result columns count.

27.33 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

```
APEX_EXEC.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,
    p_parent_column_path IN VARCHAR2 DEFAULT NULL )
RETURN PLS_INTEGER;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_attribute_label	Attribute label to format error messages.
p_column_name	Column name.
p_is_required	Indicates whether this is a required column.
p_data_type	Indicates the requested data type.
p_parent_column_path	Path to the parent column to look the index up within.

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.

27.34 GET_DATA_TYPE Function

This function converts the t_data_type constant into the VARCHAR2 representation, or the data type VARCHAR2 representation to the t_data_type constant.

Syntax

Signature 1

```
APEX_EXEC.GET_DATA_TYPE (
    p_datatype_num      IN apex_exec.t_data_type )
RETURN VARCHAR2;
```

Signature 2

```
APEX_EXEC.GET_DATA_TYPE (
    p_datatype      IN VARCHAR2 )
RETURN apex_exec.t_data_type;
```

Parameters

Parameter	Description
p_datatype_num	Data type constant of <code>apex_exec.t_data_type</code> .
p_datatype	VARCHAR2 representation of the data type, as used by SQL.

Returns

Signature 1

VARCHAR2 representation of the data type, as used by SQL

Signature 2

Data type constant of `apex_exec.t_data_type`.

27.35 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 REST Data Source contexts, the function returns the HTTP status code.

Syntax

```
APEX_EXEC.GET_DML_STATUS_CODE (
    p_context      IN t_context )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the <code>OPEN_</code> functions.

Returns

The DML status code of the current row.

27.36 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 REST Data Source contexts, the function returns the HTTP reason phrase.

Syntax

```
APEX_EXEC.GET_DML_STATUS_MESSAGE (
    p_context      IN t_context )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
<code>p_context</code>	Context object obtained with one of the <code>OPEN_</code> functions.

Returns

The DML status message of the current row.

27.37 GET_PARAMETER Functions

These functions returns a SQL parameter value. Typically used to retrieve values for `OUT` binds of an executed SQL or PL/SQL statement.

Syntax

```
APEX_EXEC.GET_PARAMETER_VARCHAR2 (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2,
    p_format_mask     IN VARCHAR2 DEFAULT NULL ) RETURN VARCHAR2;

APEX_EXEC.GET_PARAMETER_NUMBER (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN NUMBER;

APEX_EXEC.GET_PARAMETER_DATE (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN DATE;

APEX_EXEC.GET_PARAMETER_TIMESTAMP (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN TIMESTAMP;

APEX_EXEC.GET_PARAMETER_TIMESTAMP_TZ (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN TIMESTAMP WITH TIME ZONE;

APEX_EXEC.GET_PARAMETER_TIMESTAMP_LTZ (
```

```

    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN TIMESTAMP WITH LOCAL TIME ZONE;

APEX_EXEC.GET_PARAMETER_CLOB (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN CLOB;

APEX_EXEC.GET_PARAMETER_INTERVAL_D2S (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN INTERVAL DAY TO SECOND;

APEX_EXEC.GET_PARAMETER_INTERVAL_Y2M (
    p_parameters      IN t_parameters,
    p_name            IN VARCHAR2 ) RETURN INTERVAL YEAR TO MONTH;

```

Parameters

Parameter	Description
p_parameters	SQL parameter array.
p_name	Parameter name.
p_format_mask	If the parameter is of a DATE or TIMESTAMP data type, the p_format_mask parameter denotes which format mask to use to get the VARCHAR2 representation.

Returns

Parameter value.

27.38 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

```

APEX_EXEC.GET_ROW_VERSION_CHECKSUM (
    p_context      IN t_context )
RETURN VARCHAR2;

```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN functions.

Returns

The row version checksum.

27.39 GET_TOTAL_ROW_COUNT Function

This function returns the total row count of the query result.

Syntax

```
APEX_EXEC.GET_TOTAL_ROW_COUNT (
    p_context      IN t_context )
RETURN PLS_INTEGER;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

The total row count; NULL if unknown.

27.40 HAS_ERROR Function

This function returns TRUE when DML execution led to an error and FALSE when not.

Syntax

```
APEX_EXEC.HAS_ERROR (
    p_context      IN t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

TRUE if an error occurred, otherwise FALSE.

27.41 HAS_MORE_ARRAY_ROWS Function

This function returns whether the current array has more children. Can only be called within an array column; otherwise an error is raised.

Syntax

```
APEX_EXEC.HAS_MORE_ARRAY_ROWS (
    p_context      IN t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions

Returns

TRUE if successful.

FALSE if the end of the result set has been reached.

 **See Also:**

- [CLOSE_ARRAY Procedure](#)
- [OPEN_ARRAY Procedure Signature 1](#)
- [NEXT_ARRAY_ROW Function](#)

27.42 HAS_MORE_ROWS Function

This function returns whether the data source has more data after fetching p_max_rows. This function only returns a value after the NEXT_ROW loop has finished. Only then we can know that there is more data to fetch than we actually requested.

Syntax

```
APEX_EXEC.HAS_MORE_ROWS (
    p_context    IN t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

TRUE, if there is more data, FALSE otherwise. NULL if no more data detection was requested.

Examples

The following example executes a query, fetches a maximum of 10 rows, and prints the result set. If there are more rows, then a message "has more rows" displays. This example code can be used within an Execute PL/SQL region.

```
DECLARE
    l_context      apex_exec.t_context;

BEGIN
    l_context := apex_exec.open_query_context(
        p_location      => apex_exec.c_location_local_db,
        p_max_rows       => 10,
        p_sql_query     => 'select * from emp' );

    while apex_exec.next_row( l_context ) loop
```

```

        htp.p( 'EMPNO: ' || apex_exec.get_number( l_context, 'EMPNO' ) );
        htp.p( 'ENAME: ' || apex_exec.get_varchar2( l_context, 'ENAME' ) );
        htp.p( '<br>' );
    END loop;
    IF apex_exec.has_more_rows( l_context ) THEN
        htp.p( 'there are more rows ...' );
    END IF;

    apex_exec.close( l_context );
    return;
EXCEPTION
    when others then
        apex_exec.close( l_context );
        raise;
END;

```

27.43 IS_GROUP_END Function

Returns whether an end of group was found. Group columns must not be NULL.

Syntax

```
APEX_EXEC.IS_GROUP_END (
    p_context    IN t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the `OPEN_` functions.

Returns

TRUE, if successful; FALSE if no group change was found.

Examples

The following example executes a query and prints out the result set with control breaks.

```

DECLARE
    l_context          apex_exec.t_context;
    l_columns          apex_exec.t_columns;
    l_control_break    apex_exec.t_control_break;
    l_order_bys        apex_exec.t_order_bys;
BEGIN

    apex_exec.add_column(
        p_columns      => l_columns,
        p_column_name => 'EMPNO' );

    apex_exec.add_column(
        p_columns      => l_columns,
        p_column_name => 'ENAME' );

```

```

apex_exec.add_column(
    p_columns      => l_columns,
    p_column_name => 'JOB' );

apex_exec.add_column(
    p_columns      => l_columns,
    p_column_name => 'DEPTNO' );

apex_exec.add_order_by(
    p_order_bys   => l_order_bys,
    p_column_name => 'JOB',
    p_direction    => apex_exec.c_order_asc );

apex_exec.add_order_by(
    p_order_bys   => l_order_bys,
    p_column_name => 'DEPTNO',
    p_direction    => apex_exec.c_order_asc );

apex_exec.add_column(
    p_columns      => l_control_break.control_break_columns,
    p_column_name => 'JOB' );

apex_exec.add_column(
    p_columns      => l_control_break.control_break_columns,
    p_column_name => 'DEPTNO' );

l_context := apex_exec.open_query_context(
    p_location      => apex_exec.c_location_local_db,
    p_sql_query     => 'select * from emp',
    p_columns        => l_columns,
    p_order_bys     => l_order_bys,
    p_control_break => l_control_break );

WHILE apex_exec.next_row( l_context ) LOOP

    sys.dbms_output.put_line( 'EMPNO: ' || apex_exec.get_number(
        ( l_context, 'EMPNO' ) ) );
    sys.dbms_output.put_line( 'ENAME: ' || apex_exec.get_varchar2( l_context, 'ENAME' ) );
    IF apex_exec.is_group_end( p_context => l_context ) THEN
        sys.dbms_output.put_line( 'Is end of Control Break' );
    END IF;

    END LOOP;

    apex_exec.close( l_context );
    RETURN;
EXCEPTION
    WHEN others THEN
        apex_exec.close( l_context );
        RAISE;
END;

```

27.44 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

Parameter	Description
p_server_static_id	Static ID of the REST enabled SQL instance.

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.

```
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(
        p_server_static_id  => 'My_Remote_SQL' )
    then
        sys.dbms_output.put_line( 'credentials are correct!' );
    else
        sys.dbms_output.put_line( 'credentials are NOT correct!' );
    end if;
end;
```

27.45 NEXT_ARRAY_ROW Function

This function advances the array cursor by one row. Can only be called within an array column; otherwise an error is raised.

Syntax

```
APEX_EXEC.NEXT_ARRAY_ROW (
    p_context    IN t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions

Returns

TRUE if successful.

FALSE if the end of the result set has been reached.

 **See Also:**

- [OPEN_ARRAY Procedure Signature 1](#)
- [CLOSE_ARRAY Procedure](#)
- [HAS_MORE_ARRAY_ROWS Function](#)

27.46 NEXT_ROW Function

This function advances the cursor of an open query or DML context, after execution, by one row.

Syntax

```
APEX_EXEC.NEXT_ROW (
    p_context    IN t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Returns

Returns false when the end of the response has been reached, true otherwise.

27.47 OPEN_ARRAY Procedure Signature 1

Enters the array within the provided array column and moves the cursor to before the first row, so that calling next_array_row() points to the first array element.

Currently only supported for contexts on REST data sources.

Syntax

```
APEX_EXEC.OPEN_ARRAY (
    p_context          IN t_context,
    p_column_name     IN VARCHAR2 )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_column_name	Name of the array column to add a row for.

Example

```
DECLARE
    l_context apex_exec.t_context;

BEGIN
    l_context := apex_exec.open_rest_source_query(
        p_static_id      => '{REST Source static ID}',
        p_max_rows       => 1000 );

    <<rest_rows_loop>>
    WHILE apex_exec.next_row( l_context ) LOOP
        sys.dbms_output.put_line( 'ID:    ' ||
apex_exec.get_varchar2( l_context, 'TITLE' ) );
        sys.dbms_output.put_line( 'MAG:   ' ||
apex_exec.get_varchar2( l_context, 'MAG' ) );
        sys.dbms_output.put_line( 'PLACE: ' ||
apex_exec.get_varchar2( l_context, 'PLACE' ) );
        sys.dbms_output.put_line( 'TITLE: ' ||
apex_exec.get_varchar2( l_context, 'TIME' ) );
        sys.dbms_output.put_line( 'TIME:  ' ||
apex_exec.get_varchar2( l_context, 'ID' ) );

        sys.dbms_output.put_line( 'SOURCES: ' );
        apex_exec.open_array(
            p_context      => l_context,
            p_column_name  => 'SOURCES' );

        <<rest_array_row_sources_loop>>
        WHILE apex_exec.next_array_row( l_context ) LOOP

            sys.dbms_output.put_line( '-- ID:   ' ||
apex_exec.get_varchar2( l_context, 'SOURCE_ID' ) );
            sys.dbms_output.put_line( '-- NAME: ' ||
apex_exec.get_varchar2( l_context, 'SOURCE_NAME' ) );

            END LOOP rest_array_row_sources_loop;

        apex_exec.close_array( l_context );

        sys.dbms_output.put_line( 'REPORTERS: ' );
```

```

        apex_exec.open_array(
            p_context      => l_context,
            p_column_name  => 'REPORTERS' );

        <<rest_array_row_reporters_loop>>
        WHILE apex_exec.next_array_row( l_context ) LOOP

            sys.dbms_output.put_line( '-- NAME: ' ||
apex_exec.get_varchar2( l_context, 'REPORTER_NAME' ) );

            END LOOP rest_array_row_reporters_loop;

            apex_exec.close_array( l_context );

        END LOOP rest_rows_loop;

        apex_exec.close( l_context );
EXCEPTION
    WHEN others THEN
        apex_exec.close( l_context );
        RAISE;
END;

```

 **See Also:**

- [CLOSE_ARRAY Procedure](#)
- [NEXT_ARRAY_ROW Function](#)
- [SET_ARRAY_CURRENT_ROW Procedure](#)
- [ADD_DML_ARRAY_ROW Procedure](#)

27.48 OPEN_ARRAY Procedure Signature 2

Enters the array within the provided array column and moves the cursor to before the first row, so that calling `next_array_row()` points to the first array element.

Currently only supported for contexts on REST data sources.

Syntax

```
APEX_EXEC.OPEN_ARRAY (
    p_context          IN t_context,
    p_column_position IN PLS_INTEGER )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Parameter	Description
p_column_position	Position of the column to set the value for within the DML context.

See Also:

- [CLOSE_ARRAY Procedure](#)
- [NEXT_ARRAY_ROW Function](#)
- [SET_ARRAY_CURRENT_ROW Procedure](#)
- [ADD_DML_ARRAY_ROW Procedure](#)

27.49 OPEN_DUALITY_VIEW_DML_CONTEXT Function

Opens a DML context based on a Duality View source.

Syntax

```
APEX_EXEC.PROCEDUREslashFUNCTION (
    p_static_id          IN VARCHAR2,
    p_array_column_name  IN VARCHAR2
    --
    p_columns            IN t_columns
    c_empty_columns,
    p_lost_update_detection IN t_lost_update_detection DEFAULT NULL )
    RETURN t_context;
```

Parameters

Parameter	Description
p_static_id	Static ID of the Duality View source.
p_array_column_name	Name of an Array column within the REST Source data profile.
p_columns	DML columns to pass to the data source.
p_lost_update_detection	Lost-update detection type. Use constants c_lost_update_*

Returns

The context object representing the DML handle.

Example

The following example "inserts one row" into the "EMP" Duality View source.

```
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_duality_view_dml_context(
        p_static_id      => '{duality view 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;

```

27.50 OPEN_JSON_SOURCE_DML_CONTEXT Function

Opens a DML context based on a JSON source.

Syntax

```

APEX_EXEC.OPEN_JSON_SOURCE_DML_CONTEXT (
    p_static_id          IN VARCHAR2,
    p_array_column_name   IN VARCHAR2          DEFAULT NULL,
    --
    p_columns            IN t_columns          DEFAULT
c_empty_columns,
    p_lost_update_detection IN t_lost_update_detection DEFAULT NULL )
    RETURN t_context;

```

Parameters

Parameter	Description
p_static_id	Static ID of the JSON source.
p_array_column_name	Name of an Array column within the REST Source data profile.
p_columns	DML columns to pass to the data source.
p_lost_update_detection	Lost-update detection type. Use constants c_lost_update_*

Returns

The context object representing the DML handle.

Example

The following example "inserts one row" into the "EMP" Duality View source.

```

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_duality_view_dml_context(
        p_static_id           => '{duality view 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;

```

27.51 OPEN_LOCAL_DML_CONTEXT Function

This function opens a DML-context-based 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_function_body     IN VARCHAR2           DEFAULT NULL,
    p_function_body_language IN t_language   DEFAULT c_lang_plsql,
    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

Parameter	Description
p_columns	DML columns to pass to the data source.
p_query_type	Indicates the type of the data source: possible values are: <ul style="list-style-type: none"> • 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_function_body	Function body which returns the SQL query. Note that the SQL query must produce an updatable result for the DML to succeed.
p_function_body_language	Programming language used for p_function_body. Use c_lang_* constants.
p_plsql_function_body	For query type PLSQL: the PL/SQL function which returns the SQL query
p_with_check_option	Specify whether the "WITH CHECK" option should be added to the data source. If 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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.

Parameter	Description
p_sql_parameters	Bind variables to be used.

Returns

The context object representing the DML handle.

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_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_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;
```

27.52 OPEN_QUERY_CONTEXT Function Signature 1

Opens a query context for a local database, remote database, or Rest Data Source, and executes the provided SQL query.

Syntax

```

APEX_EXEC.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_match_clause               IN VARCHAR2                               DEFAULT NULL,
    p_columns_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_function_body               IN VARCHAR2                               DEFAULT NULL,
    p_function_body_language     IN t_language                            DEFAULT
c_lang_plsql,
    p_plsql_function_body        IN VARCHAR2                               DEFAULT NULL,
-- Deprecated:
    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,
    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_aggregation                 IN t_aggregation                      DEFAULT
c_empty_aggregation,
    p_control_break                IN t_control_break                    DEFAULT
c_empty_control_break,
    --
    p_post_process_type            IN t_post_processing                  DEFAULT NULL,
    --
    p_first_row                    IN NUMBER                               DEFAULT NULL,
    p_max_rows                     IN NUMBER                               DEFAULT NULL,
    --
    p_total_row_count              IN BOOLEAN                                DEFAULT FALSE,
    p_total_row_count_limit       IN NUMBER                               DEFAULT NULL,
    --
    p_supports_binary_number      IN BOOLEAN                                DEFAULT FALSE,
    --
)

```

```

    p_array_column_name      IN VARCHAR2          DEFAULT NULL,
    --
    p_duality_view_static_id IN VARCHAR2          DEFAULT NULL,
    p_json_source_static_id  IN VARCHAR2          DEFAULT NULL )
    RETURN t_context;

```

Parameters

Parameter	Description
p_location	Location to open the query context for. Can be local database, remote database, or Rest Data Source. Use c_location_* constants.
p_table_owner	Table owner when query type TABLE is used.
p_table_name	Table name when query type TABLE is used.
p_where_clause	Where clause to append when query types TABLE or GRAPH is used.
p_match_clause	Match clause to append when query type GRAPH is used.
p_columns_clause	Columns clause to append when query type GRAPH is used.
p_order_by_clause	Order by clause to append when query type TABLE is used.
p_include_rowid_column	Default FALSE. Add the ROWID column to the SELECT list when query type TABLE is used.
p_sql_query	SQL Query to execute when query type SQL Query is used.
p_function_body	Function body returning SQL query.
p_function_body_language	Programming language used for p_function_body. Use c_lang_* constants.
p_plsql_function_body	Deprecated. Use p_function_body instead.
p_optimizer_hint	Optimizer hint to be applied to the most outer SQL query generated by APEX.
p_server_static_id	Static ID of the Remote Server when REST-enabled SQL is used (such as when p_location uses c_location_remote_db).
p_module_static_id	Static ID of the REST Data Source (such as when p_location uses c_location_rest_source).
p_web_src_parameters	Parameters to be passed to a REST Data Source.
p_external_filter_expr	External filter expression to be passed to a REST Data Source.
p_external_order_by_expr	External order by expression to be passed to a REST Data Source.
p_sql_parameters	Additional bind variables to be used for the SQL query.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_columns	Columns to be selected.
p_filters	Filters to be passed to the query context.
p_order_bys	Order by expressions to be passed to the query context.
p_aggregation	Aggregation (GROUP BY, DISTINCT) to apply on top of the query.
p_control_break	Whether to return control breaks when looping through the context data.
p_post_process_type	Type of post processing to be applied to the REST Data Source result data. Use c_postprocess_* constants.
p_first_row	First row to be fetched from the result set.
p_max_rows	Maximum amount of rows to be fetched.
p_total_row_count	Whether to determine the total row count.
p_total_row_count_limit	Upper boundary for total row count computation.

Parameter	Description
p_supports_binary_number	Whether to return BINARY NUMBER columns as c_data_type_binary_number instead of c_data_type_number.
p_array_column_name	Name of an array column within the REST Source data profile.
p_duality_view_static_id	Static ID of the Duality View Source.
p_json_source_static_id	Static ID of the JSON Source.

Returns

The context object representing a "cursor" for the source 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.

```

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');
    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
        sys.dbms_output.put_line( 'EMPNO: ' || apex_exec.get_number
        ( l_context, l_idx_empno ) );
        sys.dbms_output.put_line( 'ENAME: ' ||
apex_exec.get_varchar2( l_context, l_idx_ename ) );
        sys.dbms_output.put_line( '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;

```

27.53 OPEN_QUERY_CONTEXT Function Signature 2

Enables plug-in developers to open a query context based on the current region source. All data source information that the query retrieves is from the plug-in region metadata.

Syntax

```

APEX_EXEC.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_by      DEFAULT
c_empty_order_bys,
    p_aggregation                  IN t_aggregation   DEFAULT
c_empty_aggregation,
    p_control_break                IN t_control_break DEFAULT
c_empty_control_break,
    --
    p_first_row                     IN NUMBER           DEFAULT NULL,
    p_max_rows                      IN NUMBER           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 )
    RETURN t_context;

```

Parameters

Parameter	Description
p_columns	Columns to be selected from the data source.
p_filters	Filters to be passed to the query context.
p_order_bys	Order by expressions to be passed to the query context.
p_aggregation	Aggregation (GROUP BY, DISTINCT) to apply on top of the query.
p_control_break	Whether to return control breaks when looping through the context data.
p_first_row	First row to be fetched from the result set.
p_max_rows	Maximum amount of rows to be fetched.
p_total_row_count	Whether to determine the total row count.
p_total_row_count_limit	Upper boundary for the total row count computation.
p_sql_parameters	Bind variables to be used.

Returns

The context object representing a "cursor" for the query.

27.54 OPEN_REMOTE_DML_CONTEXT Function

This function opens a DML-context-based remote database.

Syntax

```
APEX_EXEC.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_function_body           IN VARCHAR2          DEFAULT NULL,
    p_function_body_language IN t_language         DEFAULT c_lang_plsql,
    p_plsql_function_body     IN VARCHAR2          DEFAULT NULL, --
Deprecated:
    --
    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

Parameter	Description
p_server_static_id	Static ID of the ORDS REST Enabled SQL Instance.
p_columns	DML columns to pass to the Data Source.
p_query_type	Indicates the type of the Data Source. Possible values are: <ul style="list-style-type: none"> • 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.

Parameter	Description
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_function_body	Function body which returns the SQL query. Note that the SQL query must produce an updatable result for the DML to succeed.
p_function_body_language	Programming language used for p_function_body. Use c_lang_* constants.
p_plsql_function_body	Deprecated. Use p_function_body instead.
p_with_check_option	Specify whether 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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.

Returns

The context object representing the DML handle.

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(
    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;

```

27.55 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

```

APEX_EXEC.OPEN_REMOTE_SQL_QUERY (
    p_server_static_id      IN VARCHAR2,
    p_sql_query              IN VARCHAR2,
    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_first_row               IN NUMBER       DEFAULT NULL,
    p_max_rows                IN NUMBER       DEFAULT NULL,
    --
    p_total_row_count         IN BOOLEAN      DEFAULT FALSE,
    p_total_row_count_limit  IN NUMBER       DEFAULT NULL )
RETURN t_context;

```

Parameters

Parameter	Description
p_server_static_id	Static ID of the ORDS REST Enabled SQL Instance.
p_sql_query	SQL Query to execute.
p_sql_parameters	Bind variables to pass to the remote server.

Parameter	Description
p_auto_bind_items	Whether to auto-bind all page items.
p_columns	Columns to return from the SQL query.
p_first_row	First row to be fetched from the result set.
p_max_rows	Maximum amount of rows to be fetched.
p_total_row_count	Whether to determine the total row count.
p_total_row_count_limit	Upper boundary for total row count computation.

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 an "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 * from emp' );

    l_idx_empno    := apex_exec.get_column_position( l_context, 'EMPNO');
    l_idx_ename    := apex_exec.get_column_position( l_context, 'ENAME');
    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

        http.p( 'EMPNO: ' || apex_exec.get_number( l_context,
l_idx_empno ) );
        http.p( 'ENAME: ' || apex_exec.get_varchar2( l_context,
l_idx_ename ) );
        http.p( 'MGR: ' || apex_exec.get_number( l_context,
l_idx_mgr ) );
    END LOOP;
END;

```

```

        END LOOP;

        apex_exec.close( l_context );
        RETURN;
    EXCEPTION
        WHEN others THEN
            apex_debug.log_exception;
            apex_exec.close( l_context );
        RAISE;
    END;

```

27.56 OPEN_REST_SOURCE_DML_CONTEXT Function

This function opens a DML-context-based REST Data Source.

Syntax

```

FUNCTION OPEN_REST_SOURCE_DML_CONTEXT (
    p_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 )
    --
    p_fetch_rows_parameters IN t_parameters        DEFAULT
c_empty_parameters,
    p_fetch_row_parameters  IN t_parameters        DEFAULT
c_empty_parameters,
    p_insert_row_parameters IN t_parameters        DEFAULT
c_empty_parameters,
    p_update_row_parameters IN t_parameters        DEFAULT
c_empty_parameters,
    p_delete_row_parameters IN t_parameters        DEFAULT
c_empty_parameters,
    --
    p_array_column_name   IN VARCHAR2              DEFAULT NULL )
    RETURN t_context;

```

Parameters

Parameter	Description
p_static_id	Static ID of the REST Data Source to use. This REST Data Source must have operations for at least one of the Insert Rows, Update Rows or Delete rows database actions.
p_parameters	REST Data Source parameter values to pass to the DML context.
p_columns	DML columns to pass to the data source.

Parameter	Description
p_lost_update_detection	Lost-update detection type. Possible values are: <ul style="list-style-type: none"> 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_fetch_rows_parameters	REST Data Source parameter values to use only for the "Fetch Rows" operation within this DML context.
p_fetch_row_parameters	REST Data Source parameter values to use only for the "Fetch Single Row" operation within this DML context.
p_insert_row_parameters	REST Data Source parameter values to use only for the "Update" operation within this DML context.
p_update_row_parameters	REST Data Source parameter values to use only for the "Insert" operation within this DML context.
p_delete_row_parameters	REST Data Source parameter values to use only for the "Delete" operation within this DML context.
p_array_column_name	Name of an array column within the REST Source data profile.

Returns

The context object representing the DML handle.

Example

The following inserts one row into the EMP REST Data Source.

```

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_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;

```

27.57 OPEN_REST_SOURCE_QUERY Function

This function opens a REST Source query context. Based on the provided REST Source static ID, the operation matched to the `FETCH_COLLECTION` database operation will be selected.

Syntax

```

APEX_EXEC.OPEN_REST_SOURCE_QUERY (
    p_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_aggregation        IN t_aggregation     DEFAULT c_empty_aggregation,
    p_control_break      IN t_control_break   DEFAULT
    c_empty_control_break,
    p_columns             IN t_columns         DEFAULT c_empty_columns,
    --
    p_external_filter_expr IN VARCHAR2        DEFAULT NULL,
    p_external_order_by_expr IN VARCHAR2        DEFAULT NULL,
    --
    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_array_column_name   IN VARCHAR2         DEFAULT NULL )
RETURN t_context;

```

Parameters

Parameter	Description
p_static_id	Static ID of the REST Data Source to invoke.
p_parameters	Parameter values to be passed to the data source.
p_filters	Filters to be passed to the data source.
p_order_bys	Order by expressions to be passed to the data source.
p_aggregation	Aggregation (GROUP BY, DISTINCT) to apply on top of the query.
p_control_break	Whether to return control breaks when looping through the context data.
p_columns	Columns to be selected from the data source.
p_external_filter_expr	Filter expression to be passed 1:1 to the external web service. Depends on the actual web service being used.
p_external_order_by_expr	Order by expression to be passed 1:1 to the external web service. Depends on the actual web service being used.
p_first_row	First row to be fetched from the data source.

Parameter	Description
p_max_rows	Maximum amount of rows to be fetched from the data source.
p_total_row_count	Whether to determine the total row count (only supported when the attribute "allow fetch all rows" equals Yes).
p_array_column_name	Name of an array column within the REST Source data profile.

Returns

The context object representing a `cursor` for the REST Data Source query

Example

The following example assumes a REST Data Source 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.

```

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_rest_source_query(
        p_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 );

```

```
RAISE;
END;
```

27.58 OPEN_WEB_SOURCE_DML_CONTEXT Function (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use [OPEN_REST_SOURCE_DML_CONTEXT Function](#) instead.

This function opens a DML-context-based 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,
    --
    p_array_column_name     IN VARCHAR2              DEFAULT NULL )
RETURN t_context;
```

Parameters

Parameter	Description
p_module_static_id	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.
p_parameters	Web source parameter values to pass to the DML context.
p_columns	DML columns to pass to the data source
p_lost_update_detection	Lost-update detection type. Possible values are: <ul style="list-style-type: none"> • 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_array_column_name	Name of an array column within the REST Source data profile.

Returns

The context object representing the DML handle.

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;

```

27.59 OPEN_WEB_SOURCE_QUERY Function (Deprecated)

 **Note:**

This function is deprecated and will be removed in a future release. Use [OPEN_REST_SOURCE_QUERY](#) instead. See [OPEN_REST_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

```
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_aggregation         IN t_aggregation   DEFAULT c_empty_aggregation,
    p_control_break       IN t_control_break DEFAULT
c_empty_control_break,
    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,
    --
    p_array_column_name    IN VARCHAR2       DEFAULT NULL )
RETURN t_context;

```

Parameters

Parameter	Description
p_module_static_id	Static ID of the web source module to invoke.
p_parameters	Parameter values to be passed to the web source.
p_filters	Filters to be passed to the web source.
p_order_bys	Order by expressions to be passed to the web source.
p_aggregation	Aggregation (GROUP BY, DISTINCT) to apply on top of the query.
p_control_break	Whether to return control breaks when looping through the context data.
p_columns	Columns to be selected from the web source.
p_first_row	First row to be fetched from the web source.
p_max_rows	Maximum amount of rows to be fetched from the web source.
p_external_filter_expr	Filter expression to be passed 1:1 to the external web service. Depends on the actual web service being used.
p_external_order_by_expr	Order by expression to be passed 1:1 to the external web service. Depends on the actual web service being used.
p_total_row_count	Whether to determine the total row count (only supported when the attribute "allow fetch all rows" equals Yes).
p_array_column_name	Name of an array column within the REST Source data profile.

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.

```

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 );
        raise;
END;

```

27.60 PURGE_DUALITY_VIEW_CACHE Procedure

Purges the local cache for a Duality View of JSON sources based on REST-enabled SQL. The Duality View or JSON source must exist in the current application and must be identified by its static ID. If caching is disabled or no cache entries exist, nothing happens.

Syntax

```
APEX_EXEC.PURGE_DUALITY_VIEW_CACHE (
    p_static_id           IN VARCHAR2,
    p_current_session_only IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_static_id	Static ID of the JSON or Duality View source.
p_current_session_only	Default FALSE. Specify TRUE to only purge entries that were saved for the current session.

Example

```
BEGIN
    apex_exec.purge_duality_view_cache(
        p_static_id => '{Duality View static ID}' );
END;
```

27.61 PURGE_JSON_SOURCE_CACHE Procedure

Purges the local cache for a Duality View of JSON sources based on REST-enabled SQL. The Duality View or JSON source must exist in the current application and must be identified by its static ID. If caching is disabled or no cache entries exist, nothing happens.

Syntax

```
APEX_EXEC.PURGE_JSON_SOURCE_CACHE (
    p_static_id           IN VARCHAR2,
    p_current_session_only IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_static_id	Static ID of the JSON or Duality View source.
p_current_session_only	Default FALSE. Specify TRUE to only purge entries that were saved for the current session.

Example

```
BEGIN
    apex_exec.purge_duality_view_cache(
        p_static_id => '{Duality View static ID}' );
END;
```

27.62 PURGE_REST_SOURCE_CACHE Procedure

This procedure purges the local cache for a REST Data Source. The REST Data Source must exist in the current application and be identified by a static ID. If caching is disabled or no cache entries exist, nothing happens.

Syntax

```
APEX_EXEC.PURGE_REST_SOURCE_CACHE (
    p_static_id           IN VARCHAR2,
    p_current_session_only IN BOOLEAN DEFAULT FALSE )
```

Parameters

Parameter	Description
p_static_id	Static ID of the REST Data Source to invoke.
p_current_session_only	Specify TRUE to only purge entries that were saved for the current session. Default FALSE.

Example

Purge cache for the REST Data Source with static ID USGS.

```
BEGIN
    apex_exec.purge_rest_source_cache(
        p_static_id => 'USGS' );
END;
```

27.63 PURGE_WEB_SOURCE_CACHE Procedure (Deprecated)

Note:

This procedure is deprecated and will be removed in a future release. Use `purge_rest_source_cache` instead.

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

```
APEX_EXEC.PURGE_WEB_SOURCE_CACHE (
    p_module_static_id      IN VARCHAR2,
    p_current_session_only IN BOOLEAN DEFAULT FALSE )
```

Parameters

Parameter	Description
p_module_static_id	Static ID of the web source module to invoke.
p_current_session_only	Specify TRUE to only purge entries that were saved for the current session. Default FALSE.

Example

Purge cache for the Web Source Module with static ID "USGS".

```
BEGIN
    apex_exec.purge_web_source_cache(
        p_module_static_id => 'USGS' );
END;
```

27.64 SET_ARRAY_CURRENT_ROW Procedure

This procedure moves the cursor to the given row within the current array.

Currently only supported for contexts on REST Data Sources.

Syntax

```
APEX_EXEC.SET_ARRAY_CURRENT_ROW (
    p_context          IN t_context,
    p_current_row_idx  IN PLS_INTEGER )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_current_row_idx	Row within the child array to place the cursor on.

**See Also:**

- [OPEN_ARRAY Procedure Signature 1](#)
- [CLOSE_ARRAY Procedure](#)
- [NEXT_ARRAY_ROW Function](#)
- [ADD_DML_ARRAY_ROW Procedure](#)

27.65 SET_ARRAY_ROW_VERSION_CHECKSUM Procedure

This procedure sets the row version checksum for the current nested array row. Can only be called when inside an array column; otherwise an error message is called.

The checksum is to be used by a REST Data Source Plug-In, when performing plug-in actions for an array element.

Syntax

```
APEX_EXEC.SET_ARRAY_ROW_VERSION_CHECKSUM (
    p_context          IN t_context,
    p_checksum         IN VARCHAR2 );
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_checksum	Checksum to use for lost-update detection of this array row.



See Also:

- [GET_ARRAY_ROW_VERSION_CHECKSUM Function](#)

27.66 SET_CURRENT_ROW Procedure

This procedure sets the current row pointer of a DML context to the given row number. Subsequent SET_VALUE invocations affect the row with this row number.

Syntax

```
APEX_EXEC.SET_CURRENT_ROW (
    p_context    IN t_context,
    p_row_idx   IN PLS_INTEGER )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_row_idx	Row number to set the "current row" pointer to.

27.67 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

```
APEX_EXEC.SET_NULL (
    p_context          IN t_context,
    p_column_position IN PLS_INTEGER )
```

Signature 2

```
APEX_EXEC.SET_NULL (
    p_context          IN t_context,
    p_column_name     IN VARCHAR2 )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_column_position	Position of the column to set the value for within the DML context.
p_column_name	Name of the column to set the value.

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' );
```

27.68 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

```
APEX_EXEC.SET_ROW_VERSION_CHECKSUM (
    p_context          IN t_context,
    p_checksum         IN VARCHAR2 )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.

Parameter	Description
p_checksum	Checksum to use for lost-update detection of this row.

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_source_context => l_query_context );

    apex_exec.set_value(
        p_context      => l_dml_context,
        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;

```

27.69 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

Signature 1

```
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 2

```
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 3

```
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 4

```
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 5

```

PROCEDURE SET_VALUE(
    p_context          IN t_context,
    p_column_position IN PLS_INTEGER,
    p_value            IN TIMESTAMP WITH TIME ZONE);

PROCEDURE SET_VALUE(
    p_context          IN t_context,
    p_column_name     IN VARCHAR2,
    p_value            IN TIMESTAMP WITH TIME ZONE);

```

Signature 6

```

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 7

```

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 8

```

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 9

```

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 10

```

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 11

```

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 );

```

Signature 12

 **Note:**

This signature is **only** available if SDO_GEOmetry (Oracle Locator) is installed in the database.

```

PROCEDURE SET_VALUE(
    p_context      IN t_context,
    p_column_position  IN PLS_INTEGER,
    p_value        IN mdsys.sdo_geometry );

PROCEDURE SET_VALUE(
    p_context      IN t_context,
    p_column_name  IN VARCHAR2,
    p_value        IN mdsys.sdo_geometry );

```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_column_position	Position of the column to set the value for within the DML context.
p_column_name	Name of the column to set the value for.
p_value	Value to set.

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,
    p_columns_name => 'HIREDATE',
    p_value        => trunc( sysdate ) );

```

27.70 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

```
APEX_EXEC.SET_VALUES (
    p_context          IN t_context,
    p_source_context   IN t_context )
```

Parameters

Parameter	Description
p_context	Context object obtained with one of the OPEN_ functions.
p_source_context	Query context object to get column values from.

Example

See [SET_ROW_VERSION_CHECKSUM Procedure](#).

28

APEX_EXPORT

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](#)
- [UNZIP Function](#)
- [ZIP Function](#)

28.1 GET_APPLICATION Function

This function exports the given application and optionally splits the application definition into multiple files. The optional p_with_% parameters can be used to include additional information in the export.

Syntax

```
APEX_EXPORT.GET_APPLICATION (
    p_application_id           IN NUMBER,
    p_type                      IN t_export_type   DEFAULT
c_type_application_source,
    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,
    p_components                IN apex_t_varchar2 DEFAULT NULL,
    p_with_audit_info          IN t_audit_type  DEFAULT NULL,
    p_with_runtime_instances   IN apex_t_varchar2 DEFAULT NULL )
RETURN apex_t_export_files;
```

Parameters

Parameters	Description
p_application_id	The application ID.
p_split	If TRUE, splits the definition into discrete elements that can be stored in separate files. If FALSE, the result is a single file.
p_type	Comma-delimited list of export types to perform: <ul style="list-style-type: none"> • APPLICATION_SOURCE - export an APEX application using other parameters passed. • EMBEDDED_CODE - export code such as SQL, PL/SQL and JavaScript. APEX ignores all other options when EMBEDDED_CODE is selected. • CHECKSUM-SH1 - export a SHA1 checksum that is independent of IDs and can be compared across instances and workspaces. • CHECKSUM-SH256 - export a SHA-256 checksum that is independent of IDs and can be compared across instances and workspaces. • READABLE_YAML - export a readable version of the application metadata in YAML format. When the parameter p_type contains APPLICATION_SOURCE and either CHECKSUM-SH1 or CHECKSUM-SH256, APEX adds an additional line with a comment containing an overall checksum at the end of the application source file or, in case of a split export, the top-level install.sql file.
p_with_date	If TRUE, includes export date and time in the result.
p_with_ir_public_reports	If TRUE, includes public reports that a user saved.
p_with_ir_private_reports	If TRUE, includes private reports that a user saved.
p_with_ir_notifications	If TRUE, includes report notifications.
p_with_translations	If TRUE, includes application translation mappings and all text from the translation repository.
p_with_pkg_app_mapping	Note: This parameter is obsolete. If TRUE, exports installed packaged applications with references to the packaged application definition. If FALSE, exports them as normal applications.
p_with_original_ids	If TRUE, exports with the IDs as they were when the application was imported.
p_with_no_subscriptions	If FALSE, components contain subscription references.
p_with_comments	If TRUE, includes developer comments.
p_with_supporting_objects	If Y, exports supporting objects. If I, installs on import automatically. If N, does not export supporting objects. If NULL, uses the application's include in export deployment value.
p_with_acl_assignments	If TRUE, exports ACL user role assignments.

Parameters	Description
p_components	If not NULL, exports only given components (array elements should be of form <i>type:name</i> , for example, PAGE:42 or MESSAGE:12345). See view APEX_APPL_EXPORT_COMPS for components that can be exported.
p_with_audit_info	Use % to indicate that all components of the given type should be exported. For example: LOV:% exports all Lists Of Values contained in the application.
p_with_runtime_instances	Specifies the detail of audit information to include: <ul style="list-style-type: none"> • NULL: export excludes all audit information. • NAMES_AND_DATES: export includes Created On, Created By, Updated On, Updated By values if they exist. • DATES_ONLY: export includes Created On and Updated On values if they exist. User names are excluded. An array with components for which to export runtime instance data. For example, specify WORKFLOW to export all Workflow Instances together with the application.

Returns

A table of apex_t_export_file. Unless the caller passes p_split=>true to the function, the result is a single file.

Example 1

This SQLcl 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
```

Example 2

The following example shows an install.sql file that was exported with p_type => 'APPLICATION_SOURCE,CHECKSUM-SH1'

```
prompt --install
@@application/set_environment.sql
```

```

@@application/delete_application.sql
...snip...
@@application/deployment/buildoptions.sql
@@application/end_environment.sql
-- Application Checksum SH1:jpclimUZZDVB1MKpyyAfPBDww=

```

28.2 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 apex_t_export_files;

```

Parameters

Parameters	Description
p_workspace_id	The workspace ID.
p_with_date	If TRUE, include export date and time in the result.

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_files(p_workspace_id => 12345678);
END;

```

28.3 GET_FEEDBACK Function

This function exports user feedback to the development environment or developer feedback to the deployment environment.

Syntax

```

APEX_EXPORT.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 apex_t_export_files;

```

Parameters

Parameters	Description
p_workspace_id	The workspace id.
p_with_date	If TRUE, include export date and time in the result.
p_since	If set, only export feedback that has been gathered since the given date.
p_deployment_system	If NULL, export user feedback. If not NULL, export developer feedback for the given deployment system.

Returns

A table of apex_t_export_file.

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;
```

28.4 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

```
APEX_EXPORT.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 apex_t_export_files;
```

Parameters

Parameters	Description
p_workspace_id	The workspace ID.
p_with_date	If TRUE, include export date and time in the result.
p_with_team_development	If TRUE, include team development data.
p_with_misc	If TRUE, include data from SQL Workshop, mail logs, and so on, in the export.

Returns

A table of `apex_t_export_file`.

Examples

The following example exports the definition of workspace #12345678.

```
DECLARE
    l_file apex_t_export_files;
BEGIN
    l_files := apex_export.get_workspace(p_workspace_id => 12345678);
END;
```

28.5 UNZIP Function

This function extracts and decompresses all the files from a zip archive.

This function is intended for use with the routines in the `APEX_APPLICATION_INSTALL` package and assumes that all of the files in the ZIP archive are in a text format, such as SQL scripts (which must have a `.sql` extension) or simple README files.

All text content in the ZIP file must be encoded as UTF-8.

Syntax

```
APEX_EXPORT.UNZIP (
    p_source_zip      IN BLOB )
    RETURN apex_t_export_files;
```

Parameters

Parameter	Description
p_source_zip	A BLOB containing the zip archive.

Returns

This function returns a table of `apex_t_export_file` containing the name and contents (converted to text format) of each file from the ZIP archive.

Example

The following example fetches an application archive from a remote URL, extracts the files within it, and prints the type and name of the contained application.

```

DECLARE
    l_zip blob;
    l_info apex_application_install.t_file_info;
BEGIN
    l_zip := apex_web_service.make_rest_request_b (
        p_url => 'https://www.example.com/apps/f100.zip',
        p_http_method => 'GET' );
    l_info := apex_application_install.get_info (
        p_source => apex_export.unzip (
            p_source_zip => l_zip ) );

    sys.dbms_output.put_line (
        apex_string.format (
            p_message => q'~
                !Type ..... %0
                !App Name ..... %1
                !~',
            p0 => l_info.file_type,
            p1 => l_info.app_name,
            p_prefix => '!' ) );
END;

```

28.6 ZIP Function

This function compresses a list of files (usually obtained from one of the `APEX_EXPORT` routines) into a single BLOB containing a `.zip` archive. All text content in the resultant `.zip` file is encoded as UTF-8.

All file names within the archive must be unique to prevent the accidental overwriting of files in the application export (an exception raises otherwise).

Additional files (`p_extra_files`) may also be added to the resultant archive, such as a simple `README.txt` file or licensing information.

Syntax

```

APEX_EXPORT.ZIP (
    p_source_files    apex_t_export_files,
    p_extra_files     apex_t_export_files DEFAULT apex_t_export_files() )
RETURN BLOB;

```

Parameters

Parameter	Description
<code>p_source_files</code>	A table of files. For example, from <code>apex_export.get_application</code> .
<code>p_extra_files</code>	Optional additional files to add to the resultant <code>.zip</code> archive.

Returns

This function returns a BLOB containing the compressed application files and any extra files, in ZIP format.

Example

```
DECLARE
    l_source_files apex_t_export_files;
    l_extra_files apex_t_export_files;
    l_zip blob;
BEGIN
    l_source_files := apex_export.get_application(
        p_application_id => 100,
        p_split => true );

    l_extra_files := apex_t_export_files(
        apex_t_export_file(
            name => 'README.md',
            contents => 'An example exported application.' ),
        apex_t_export_file(
            name => 'LICENSE.txt',
            contents => 'The Universal Permissive License (UPL), Version 1.0' ) );

    l_zip := apex_export.zip(
        p_source_files => l_source_files,
        p_extra_files => l_extra_files );

    sys.dbms_output.put_line(
        'Compressed application export to zip of size; '
        || sys.dbms_lob.getLength( l_zip ) );
END;
```

APEX_EXTENSION

The APEX_EXTENSION package contains utility functions used for invoking extension applications.

This API can be used in the following contexts:

- in an Oracle APEX session context of a workspace that has the Component Availability attribute `Allow Hosting Extensions` enabled (is an extension workspace)
- this extension workspace has created links in its Extension Menu with attribute `public` set to Yes
- another workspace is subscribed to the extension workspace's published extension menu and granted read access to the extension workspace

The API can be called in Automations or in database sessions using `APEX_SESSION.CREATE_SESSION` to establish an APEX session context. Invoking the procedure from the extension workspace, with the subscribed workspace name or ID, from an automation of an application in that extension workspace or session in a database schema associated to it, changes the behavior of application-related public APEX views such that querying them returns the application metadata of that subscribed workspace, but not the metadata of the "own" workspace anymore.

- [ADD_MENU_ENTRY Procedure](#)
- [GET_GRANTOR_WORKSPACE Function](#)
- [REMOVE_MENU_ENTRY Procedure](#)
- [SET_WORKSPACE Procedure Signature 1](#)
- [SET_WORKSPACE Procedure Signature 2](#)

29.1 ADD_MENU_ENTRY Procedure

This procedure adds a builder extension menu link. Requires the `APEX_ADMINISTRATOR_ROLE`.

Syntax

```
APEX_EXTENSION.ADD_MENU_ENTRY (
    p_label          IN VARCHAR2,
    p_url            IN VARCHAR2,
    p_display_sequence IN NUMBER    DEFAULT NULL,
    p_description     IN VARCHAR2  DEFAULT NULL,
    p_is_public       IN BOOLEAN   DEFAULT FALSE,
    p_workspace        IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_label	Menu entry label. Must be unique within a workspace.
p_url	The menu entry's URL.
p_display_sequence	(Optional) Display sequence for sorting menu entry. Default NULL: the value is calculated and the entry is appended as last.
p_description	(Optional) Description.
p_is_public	Default FALSE. If TRUE, the entry is available for subscribing workspaces. The value TRUE can only be set for extension workspaces.
p_workspace	If the given workspace is not enabled for hosting extensions, the flag is set to FALSE. Default NULL, which means the menu entry is created for the current workspace. Value can be set to any existing workspace name.

Example

The following example adds an extension menu link in workspace MY_WORKSPACE with label "Example."

```
BEGIN
    apex_extension.add_menu_entry(
        p_label      => 'Example',
        p_url        => 'https://example.com'
        p_description => 'This is an example'
        p_workspace   => 'MY_WORKSPACE' );
END;
```

29.2 GET_GRANTOR_WORKSPACE Function

This function gets current grantor workspace name.

Syntax

```
APEX_EXTENSION.GET_GRANTOR_WORKSPACE
    RETURN VARCHAR2;
```

Parameters

None.

Returns

Workspace name of grantor workspace.

Example

The following example query returns the name of the invoking workspace in a builder extension context.

```
select apex_extension.get_grantor_workspace from sys.dual;
```

29.3 REMOVE_MENU_ENTRY Procedure

This procedure removes an existing builder extension menu link entry. Requires the APEX_ADMINISTRATOR_ROLE.

Syntax

```
APEX_EXTENSION.REMOVE_MENU_ENTRY (
    p_label      IN VARCHAR2,
    p_workspace  IN VARCHAR2  DEFAULT NULL )
```

Parameters

Parameter	Description
p_label	Menu entry label.
p_workspace	Default NULL, which means the menu entry is from the current workspace. Value can be set to any existing workspace name.

Example

The following example deletes the builder extension menu entry with label "Example" in the current workspace.

```
BEGIN
    apex_extension.remove_menu_entry(p_label => 'Example');
END;
```

29.4 SET_WORKSPACE Procedure Signature 1

This procedure sets the current workspace to the workspace that is processed by the extension application or background automation by its ID.

After calling this API, all Oracle APEX dictionary views show the metadata of that workspace.

Syntax

```
APEX_EXTENSION.SET_WORKSPACE (
    p_id      IN NUMBER )
```

Parameters

Parameter	Description
p_id	The ID of the workspace to be accessed.

Example

The following example sets access for an extension application to workspace with 123456789.

```
BEGIN
    apex_extension.set_workspace( p_id => 123456789 );
END;
```

29.5 SET_WORKSPACE Procedure Signature 2

This procedure sets the current workspace to the workspace that is processed by the extension application or background automation by its name.

After calling this API, all Oracle APEX dictionary views show the metadata of that workspace.

Syntax

```
APEX_EXTENSION.SET_WORKSPACE (
    p_name  IN VARCHAR2 )
```

Parameters

Parameter	Description
p_name	The (display) name of the workspace to be accessed.

Example

The following example sets access for an extension application to workspace MYWORKSPACE.

```
BEGIN
    apex_extension.set_workspace( p_name => 'MYWORKSPACE' );
END;
```

APEX_HTTP

The APEX_HTTP package provides APIs to download files.

- [DOWNLOAD Procedure Signature 1](#)
- [DOWNLOAD Procedure Signature 2](#)

30.1 DOWNLOAD Procedure Signature 1

This procedure downloads a BLOB to the client.

 **Note:**

Clears any previous output to the HTP buffer. `APEX_APPLICATION.STOP_APEX_ENGINE` is called after downloading the file.

Syntax

```
APEX_HTTP.DOWNLOAD (
    p_blob          IN OUT NOCOPY  BLOB,
    p_content_type IN             VARCHAR2,
    p_filename     IN             VARCHAR2  DEFAULT NULL,
    p_is_inline    IN             BOOLEAN   DEFAULT FALSE )
```

Parameters

Parameter	Description
p_blob	The BLOB value to download.
p_content_type	The mime type of the file.
p_filename	Name of the file.
p_is_inline	If FALSE (default), the browser displays a file download dialog to save the file. If TRUE, displays the file inline in the browser window.

Example

The following example downloads a file stored in a table.

```
DECLARE
    l_file          blob;
    l_content_type varchar2( 4000 );
    l_filename     varchar2( 4000 );
BEGIN
    SELECT blob_content,
```

```

        mime_type,
        filename
    INTO l_file,
        l_content_type,
        l_filename
    FROM apex_application_temp_files
    WHERE name = :P1_FILE;

apex_http.download(
    p_blob          => l_file,
    p_content_type  => l_content_type,
    p_filename      => l_filename );

END;

```

30.2 DOWNLOAD Procedure Signature 2

This procedure downloads a CLOB to the client.

 **Note:**

Clears any previous output to the HTP buffer. `APEX_APPLICATION.STOP_APEX_ENGINE` is called after downloading the file.

Syntax

```
APEX_HTTP.DOWNLOAD (
    p_clob          IN OUT NOCOPY  CLOB,
    p_content_type  IN          VARCHAR2,
    p_filename      IN          VARCHAR2 DEFAULT NULL,
    p_is_inline     IN          BOOLEAN DEFAULT FALSE )
```

Parameters

Parameter	Description
p_clob	The CLOB value to download.
p_content_type	The mime type of the file.
p_filename	Name of the file.
p_is_inline	If FALSE (default), the browser displays a file download dialog to save the file. If TRUE, displays the file inline in the browser window.

Example

The following example downloads a text.

```

DECLARE
    l_text    clob;
BEGIN
```

```
l_text := 'Hello World';

apex_http.download(
    p_clob      => l_text,
    p_content_type => 'text/plain',
    p_filename   => 'hello.txt' );

END;
```

APEX_HUMAN_TASK

The APEX_HUMAN_TASK package contains supporting APIs for the Human Task sub-feature of Approvals.

- Constants and Data Types
- ADD_TASK_COMMENT Procedure
- ADD_TASK_POTENTIAL_OWNER Procedure
- ADD_TO_HISTORY Procedure
- APPROVE_TASK Procedure
- CANCEL_TASK Procedure
- CLAIM_TASK Procedure
- COMPLETE_TASK Procedure
- CREATE_TASK Function
- DELEGATE_TASK Procedure
- GET_LOV_PRIORITY Function
- GET_LOV_STATE Function
- GET_LOV_TYPE Function
- GET_NEXT_PURGE_TIMESTAMP Function
- GET_TASK_DELEGATES Function
- GET_TASK_HISTORY Function
- GET_TASK_PARAMETER_OLD_VALUE Function
- GET_TASK_PARAMETER_VALUE Function
- GET_TASK_PRIORITIES Function
- GET_TASKS Function
- HANDLE_TASK_DEADLINES Procedure
- HAS_TASK_PARAM_CHANGED Function
- IS_ALLOWED Function
- IS_BUSINESS_ADMIN Function
- IS_OF_PARTICIPANT_TYPE Function
- REJECT_TASK Procedure
- RELEASE_TASK Procedure
- REMOVE_POTENTIAL_OWNER Procedure
- RENEW_TASK Function
- REQUEST_MORE_INFORMATION Procedure
- SET_INITIATOR_CAN_COMPLETE Procedure

- [SET_TASK_DUE Procedure](#)
- [SET_TASK_PARAMETER_VALUES Procedure](#)
- [SET_TASK_PRIORITY Procedure](#)
- [SUBMIT_INFORMATION Procedure](#)

31.1 Constants and Data Types

The APEX_HUMAN_TASK package uses the following constants and data types.

Task Types

<code>c_task_type_approval</code>	<code>constant t_task_type := 'APPROVAL';</code>
<code>c_task_type_action</code>	<code>constant t_task_type := 'ACTION';</code>

Task List Context Types

<code>c_context_my_tasks</code>	<code>constant t_task_list_context := 'MY_TASKS';</code>
<code>c_context_admin_tasks</code>	<code>constant t_task_list_context := 'ADMIN_TASKS';</code>
<code>c_context_initiated_by_me</code>	<code>constant t_task_list_context :=</code>
<code>'INITIATED_BY_ME';</code>	
<code>c_context_single_task</code>	<code>constant t_task_list_context := 'SINGLE_TASK';</code>

Task Definition Participant Types

<code>c_task_potential_owner</code>	<code>constant t_task_participant_type :=</code>
<code>'POTENTIAL_OWNER';</code>	
<code>c_task_business_admin</code>	<code>constant t_task_participant_type :=</code>
<code>'BUSINESS_ADMIN';</code>	

Task Definition Participant Identity Types

<code>c_task_identity_type_user</code>	<code>constant t_task_identity_type := 'USER';</code>
--	---

Task (Instance) Priority Constants

<code>c_task_priority_lowest</code>	<code>constant integer := 5;</code>
<code>c_task_priority_low</code>	<code>constant integer := 4;</code>
<code>c_task_priority_medium</code>	<code>constant integer := 3;</code>
<code>c_task_priority_high</code>	<code>constant integer := 2;</code>
<code>c_task_priority_urgent</code>	<code>constant integer := 1;</code>

Task (Instance) States

<code>c_task_state_unassigned</code>	<code>constant t_task_state := 'UNASSIGNED';</code>
<code>c_task_state_assigned</code>	<code>constant t_task_state := 'ASSIGNED';</code>
<code>c_task_state_completed</code>	<code>constant t_task_state := 'COMPLETED';</code>
<code>c_task_state_cancelled</code>	<code>constant t_task_state := 'CANCELLED';</code>
<code>c_task_state_failed</code>	<code>constant t_task_state := 'FAILED';</code>
<code>c_task_state_errorred</code>	<code>constant t_task_state := 'ERRORRED';</code>

```
c_task_state_expired      constant t_task_state := 'EXPIRED';
c_task_state_info_requested constant t_task_state := 'INFO_REQUESTED';
```

Task (Instance) Outcomes

```
c_task_outcome_approved   constant t_task_outcome := 'APPROVED';
c_task_outcome_rejected   constant t_task_outcome := 'REJECTED';
```

Task (Instance) Operations

```
c_task_op_approve         constant t_task_operation := 'APPROVE_TASK';
c_task_op_reject          constant t_task_operation := 'REJECT_TASK';
c_task_op_complete        constant t_task_operation := 'COMPLETE_TASK';
c_task_op_claim           constant t_task_operation := 'CLAIM_TASK';
c_task_op_delegate        constant t_task_operation := 'DELEGATE_TASK';
c_task_op_renew           constant t_task_operation := 'RENEW_TASK';
c_task_op_release         constant t_task_operation := 'RELEASE_TASK';
c_task_op_cancel          constant t_task_operation := 'CANCEL_TASK';
c_task_op_set_priority    constant t_task_operation := 'SET_TASK_PRIORITY';
c_task_op_add_comment     constant t_task_operation := 'ADD_TASK_COMMENT';
c_task_op_add_owner       constant t_task_operation :=
'ADD_TASK_POTENTIAL_OWNER';
c_task_op_request_info    constant t_task_operation := 'REQUEST_INFO';
c_task_op_submit_info     constant t_task_operation := 'SUBMIT_INFO';
c_task_op_set_due_date    constant t_task_operation := 'SET_DUE_DATE';
c_task_op_remove_owner    constant t_task_operation :=
'REMOVE_POTENTIAL_OWNER';
c_task_op_set_params       constant t_task_operation := 'SET_TASK_PARAMS';
```

Task (Instance) date formats

```
c_canonical_date_format  constant varchar2(16)      := 'YYYYMMDDHH24MISS';
```

Task Parameters Default

```
c_empty_task_parameters t_task_parameters;
```

Global Data Types

```
subtype t_task_participant_type is varchar2(15);
subtype t_task_identity_type   is varchar2(32);
subtype t_task_type            is varchar2(32);
subtype t_task_outcome         is varchar2(32);
subtype t_task_state           is varchar2(15);
subtype t_task_operation       is varchar2(30);
subtype t_task_list_context    is varchar2(15);
```

Data Types

Task Parameter (Value)

Attribute	Description
static_id	The static ID of the parameter. This ID must match the static ID of the corresponding parameter in the task definition.
string_value	The value of the parameter as a string.

```
type t_task_parameter is record (
    static_id          varchar2(32767),
    string_value       varchar2(32767)
);
```

Collection of Task Parameter Values

```
type t_task_parameters is table of t_task_parameter index by pls_integer;
```

Collection of Task Participant Types

```
type t_task_participant_types is table of t_task_participant_type
index by pls_integer;
```

31.2 ADD_TASK_COMMENT Procedure

This procedure adds a comment to a task. Any potential owner or business administrator of a Task can add comments to a Task. Comments are useful as additional information regarding a Task. For example, a manager may add her notes to a Task she is working on before delegating the Task.

Syntax

```
APEX_HUMAN_TASK.ADD_TASK_COMMENT (
    p_task_id           IN NUMBER,
    p_text              IN VARCHAR2 );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_text	The comment text.

Example

```
BEGIN
    add_task_comment(
        p_task_id => 1234,
        p_text      => 'Please review and approve');
END;
```

31.3 ADD_TASK_POTENTIAL_OWNER Procedure

This procedure adds a new potential owner to a task. Only a Business Administrator for the task can invoke this procedure. The procedure throws an error if the task is in Completed or Errorred state.

Syntax

```
APEX_HUMAN_TASK.ADD_TASK_POTENTIAL_OWNER (
    p_task_id           IN NUMBER,
    p_potential_owner   IN VARCHAR2,
    p_identity_type     IN t_task_identity_type default
c_task_identity_type_user );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_potential_owner	The potential owner.
p_identity_type	The identity type of the potential owner. Default is USER.

Note:

As of this release, the only supported identity type is USER. Additional options will be added in a future release.

Example

The following example adds user STIGER as potential owner for Task ID 1234.

```
BEGIN
    apex_human_task.add_task_potential_owner(
        p_task_id      => 1234,
        p_potential_owner => 'STIGER'
    );
END;
```

31.4 ADD_TO_HISTORY Procedure

This procedure adds a log entry into the task history and is to be used within task action code.

Syntax

```
APEX_HUMAN_TASK.ADD_TO_HISTORY (
    p_message IN VARCHAR2 )
```

Parameters

Parameter	Description
p_message	Message to add into to the task history.

Example

The following example demonstrates how to write log information. The task action uses `select * from emp` as the action source query.

```
BEGIN
    apex_human_task.add_to_history(
        p_message => 'Approved leave for employee with empno: ' || :EMPNO );
    my_logic_package.update_emp_leave_balance(
        p_empno      => :EMPNO,
        p_no_of_days  => :NO_OF_DAYS);
END;
```

31.5 APPROVE_TASK Procedure

This procedure approves a Task. Only the potential owner or actual owner of the task can invoke this procedure. This procedure moves the state of the Task to `Completed` and sets the outcome of the Task to `Approved`.

This is a convenience procedure and equivalent to calling `complete_task` with outcome `apex_approval.c_task_outcome_approved`.

Syntax

```
APEX_HUMAN_TASK.APPROVE_TASK (
    p_task_id           IN NUMBER,
    p_autoclaim         IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_autoclaim	If Task is in state UNASSIGNED then claims the task implicitly.

State Handling

Pre-State: ASSIGNED|UNASSIGNED (p_autoclaim=true)

Post-State: COMPLETED

Example

```
BEGIN
    apex_human_task.approve_task(
        p_task_id => 1234);
END;
```

31.6 CANCEL_TASK Procedure

This procedure cancels the task by setting the task to state CANCELED. Only the initiator or the Business Administrator of the task can invoke this procedure. Only tasks which are not in COMPLETED or ERRORED state can be CANCELED.

Canceling a task is useful when an approval is no longer required. For example, consider a travel approval for a business trip, and the person requesting the approval suddenly cannot make the trip, and the Task may be canceled.

Syntax

```
APEX_HUMAN_TASK.CANCEL_TASK (
    p_task_id           IN NUMBER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.

State Handling

Pre-State: Any

Post-State: CANCELED

Example

```
BEGIN
    apex_human_task.cancel_task(
        p_task_id => 1234
    );
END;
```

31.7 CLAIM_TASK Procedure

This procedure claims responsibility for a task. A task can be claimed by potential owners of the Task. A Task must be in "Unassigned" state to claim it. Once the task is claimed by a user, the Task transitions to "Assigned" state and the actual owner of the task is set to the user who claimed the task.

Syntax

```
APEX_HUMAN_TASK.CLAIM_TASK (
    p_task_id           IN NUMBER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.

State Handling

Pre-State: UNASSIGNED. Post-State: ASSIGNED.

Example

```
BEGIN
    apex_human_task.claim_task(
        p_task_id => 1234);
END;
```

31.8 COMPLETE_TASK Procedure

This procedure completes a task with an outcome. Only the actual owner or a potential owner of the task can invoke this procedure.

Tasks in Assigned state might be completed with an outcome. This operation transitions the Task from Assigned state to Completed state and sets the outcome of the task. Once a Task is in Completed state, it is subject for purging and archival.

Syntax

```
APEX_HUMAN_TASK.COMPLETE_TASK (
    p_task_id           IN NUMBER,
    p_outcome          IN t_task_outcome DEFAULT NULL,
    p_autoclaim        IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_outcome	The outcome of the Task for Approval Tasks.
p_autoclaim	If Task is in state UNASSIGNED then claim the task implicitly.

State Handling

Pre-State: ASSIGNED | UNASSIGNED (p_autoclaim=true)

Post-State: COMPLETED

Example

```
BEGIN
    apex_human_task.complete_task(
        p_task_id => 1234,
        p_outcome => apex_human_task.c_task_outcome_approved
    );
END;
```

31.9 CREATE_TASK Function

This function creates a new task. A new Task (Instance) is created. Depending on the task definition participant setting, the Task is set to state Unassigned or Assigned.

If the task definition has a single potential owner, the Task is set to Assigned.

If the task has multiple potential owners, the Task is set to Unassigned and can be claimed by any of the potential owners. This procedure throws an exception if no potential owners are found in the corresponding task definition.

Syntax

```
APEX_HUMAN_TASK.CREATE_TASK (
    p_application_id           IN NUMBER              DEFAULT
apex_application.g_flow_id,
    p_task_def_static_id       IN VARCHAR2,
    p_subject                  IN VARCHAR2          DEFAULT NULL,
    p_parameters               IN t_task_parameters DEFAULT
c_empty_task_parameters,
    p_priority                 IN INTEGER             DEFAULT NULL,
    p_initiator                IN VARCHAR2          DEFAULT NULL,
    p_initiator_can_complete   IN BOOLEAN            DEFAULT NULL,
    p_detail_pk                IN VARCHAR2          DEFAULT NULL,
    p_due_date                 IN TIMESTAMP WITH TIME ZONE DEFAULT NULL )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_application_id	The application ID that creates the Task.
p_task_def_static_id	The Task Definition static ID.
p_subject	The subject (expression of the Task).
p_parameters	The task parameters.
p_priority	(Optional) A task priority, default is NULL. If no priority is provided, uses the priority set in the corresponding task definition.
p_initiator	(Optional) If TRUE, the initiator of the task instance can approve or reject this task. If no value is provided, the setting in the corresponding task definition is used.
p_initiator_can_complete	(Optional) Enables the initiator of a task to complete the task (default NULL). If this parameter is not specified, the value of the corresponding task definition is used.
p_detail_pk	(Optional) A primary key value for the task details.
p_due_date	(Optional) Page Item representing the Due Date of the Task. When specified, this value overrides the Due Date provided in the Task Definition this Task is based on.

Returns

Returns the ID of the newly created task.

Example

The following example creates a requisition item in the system of record in the database and then creates a new Human Task to get the requisition item approved by a user.

```

DECLARE
    l_req_id      number;
    l_req_item    varchar2(100) := 'Some requisition item requiring approval';
    l_req_amount  number := 2499.42;
    l_task_id     number;
BEGIN
    insert into requisitions(created_by, creator_emailid, item, item_amount,
item_category)
        values (:emp_uid, :emp_email, l_req_item, l_req_amount, 'Equipment')
        returning id into l_req_id;
    commit;

    l_task_id := apex_approval.create_task(
        p_application_id => 110,
        p_task_def_static_id => 'REQAPPROVALS',
        p_subject => 'Requisition ' || l_req_id || ':' || l_req_item || '
for ' || l_req_amount,
        p_initiator => :emp_uid,
        p_initiator_can_complete => true,
        p_parameters => apex_approval.t_task_parameters(
            1 => apex_approval.t_task_parameter(static_id => 'REQ_DATE',
string_value => sysdate),
            3 => apex_approval.t_task_parameter(static_id => 'REQ_AMOUNT',
string_value => l_req_amount),
            4 => apex_approval.t_task_parameter(static_id => 'REQ_ITEM',
string_value => l_req_item),
            5 => apex_approval.t_task_parameter(static_id => 'REQ_ID',
string_value => l_req_id)
        ),
        p_detail_pk => l_req_id
    );
END;

```

31.10 DELEGATE_TASK Procedure

This procedure assigns the task to one potential owner and sets the task state to `Assigned`. Either the current owner of the task (the user to whom the task is currently assigned) or the Business Administrator of the task can perform this operation.

Syntax

```

APEX_HUMAN_TASK.DELEGATE_TASK (
    p_task_id          IN NUMBER,
    p_to_user          IN VARCHAR2 );

```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_to_user	A (user) participant.

State Handling

Pre-State: UNASSIGNED, ASSIGNED

Post-State: ASSIGNED

Example

```
BEGIN
    apex_human_task.delegate_task(
        p_task_id          => 1234,
        p_to_user          => 'STIGER'
    );
END;
```

31.11 GET_LOV_PRIORITY Function

This function retrieves the list of value data for the task priority.

Syntax

```
APEX_HUMAN_TASK.GET_LOV_PRIORITY
RETURN wwv_flow_t_temp_lov_data pipelined;
```

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,val from table ( apex_human_task.get_lov_priority )
```

31.12 GET_LOV_STATE Function

This function gets the list of value data for the task attribute state.

Syntax

```
APEX_HUMAN_TASK.GET_LOV_STATE
RETURN wwv_flow_t_temp_lov_data pipelined;
```

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,val from table ( apex_human_task.get_lov_state )
```

31.13 GET_LOV_TYPE Function

Gets the list of value data for the task attribute type.

Syntax

```
APEX_HUMAN_TASK.GET_LOV_TYPE
RETURN apex_t_temp_lov_data;
```

Parameters

None.

Returns

apex_t_temp_lov_data

Example

```
select disp,
       val
  from table ( apex_approval.get_lov_type )
```

31.14 GET_NEXT_PURGE_TIMESTAMP Function

This function retrieves the timestamp of the next purge.

Syntax

```
APEX_HUMAN_TASK.GET_NEXT_PURGE_TIMESTAMP
RETURN timestamp with time zone;
```

Parameters

None.

Returns

Returns the timestamp of the next purge.

Example

```
DECLARE
  l_next_purge_job_ts timestamp with time zone;
BEGIN
  l_next_purge_job_ts := apex_human_task.get_next_purge_timestamp();
END;
```

31.15 GET_TASK_DELEGATES Function

This function gets the potential new owners of a task. The actual owner is excluded from the list.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_HUMAN_TASK.GET_TASK_DELEGATES (
    p_task_id IN NUMBER )
RETURN wwv_flow_t_temp_lov_data pipelined;
```

Parameters

Parameter	Description
p_task_id	The task ID.

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,val from table ( apex_human_task.get_task_delegates ( p_task_id
=> 1234 ) )
```

31.16 GET_TASK_HISTORY Function

This function gets the approval log for a task.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_HUMAN_TASK.GET_TASK_HISTORY (
    p_task_id      IN NUMBER,
    p_include_all  IN VARCHAR2 DEFAULT 'N' )
RETURN wwv_flow_t_approval_log_table pipelined;
```

Parameters

Parameter	Description
p_task_id	The task ID.
p_include_all	If set to Y, the history of all tasks linked to the task with the given task ID is shown. In 22.2, this includes prior Tasks that have been expired.

Returns

A table of approval log entries (type apex_t_approval_log).

Example

```
select *  from table ( apex_human_error.get_task_history ( p_task_id => 1234,
                                                               p_include_all => 'Y' ) )
```

31.17 GET_TASK_PARAMETER_OLD_VALUE Function

This function retrieves the old value of a parameter of this task that was updated in the current session. Raises a "No Data Found" error if the parameter does not exist and `p_raise_error` flag is set to TRUE.

Syntax

```
APEX_HUMAN_TASK.GET_TASK_PARAMETER_OLD_VALUE (
    p_task_id          IN NUMBER,
    p_param_static_id  IN VARCHAR2,
    p_raise_error      IN BOOLEAN DEFAULT TRUE )
```

Parameters

Parameter	Description
<code>p_task_id</code>	The Task ID.
<code>p_param_static_id</code>	The static ID of the parameter.
<code>p_raise_error</code>	If TRUE, raises an error if the parameter is not found.

Returns

VARCHAR2 - The old value of this parameter in VARCHAR2 format.

Example

```
BEGIN
    return apex_human_task.get_task_parameter_old_value(
        p_task_id          => 1234,
        p_param_static_id  => 'REQ_AMOUNT',
        p_raise_error      => false);
END;
```

31.18 GET_TASK_PARAMETER_VALUE Function

This function gets the value of a Task parameter. This function can be used in SQL or PL/SQL to get the value of a Task parameter for a given task.

Syntax

```
APEX_HUMAN_TASK.GET_TASK_PARAMETER_VALUE (
    p_task_id          IN NUMBER,
    p_param_static_id IN VARCHAR2,
    p_ignore_not_found IN BOOLEAN DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_param_static_id	The static ID of the parameter.
p_ignore_not_found	If set to FALSE (default) and no data is found, a no_data_found exception raises. If set to TRUE and no data is found, returns NULL.

Returns

The task parameter value for the given static ID or null.

Exception

no_data_found - In the case where p_ignore_not_found is set to false and no data is found (for example, if the parameter of given name does not exist).

Example

```
DECLARE
    l_req_item varchar2(100);
BEGIN
    l_req_item := apex_human_task.get_task_parameter_value(
        p_task_id      => 1234,
        p_param_static_id => 'REQ_ITEM'
    );
    dbms_output.put_line('Parameter REQ_ITEM of task 1234 has value ' ||
    l_req_item);
END;
```

31.19 GET_TASK_PRIORITIES Function

This function gets the potential new priorities of a task. The actual priority is excluded from the list.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_HUMAN_TASK.GET_TASK_PRIORITIES (
    p_task_id IN NUMBER )
RETURN apex_t_temp_lov_data pipelined;
```

Parameters

Parameter	Description
p_task_id	The task ID.

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,val from table ( apex_human_task.get_task_priorities ( p_task_id
=> 1234 ) )
```

31.20 GET_TASKS Function

This function gets the tasks of a user depending on the given context.

Context can be one of the following:

- MY_TASKS - Returns all tasks where the user calling the function is either the Owner or one of the Potential Owners of the task.
- ADMIN_TASKS - Returns all tasks for which the user calling the function is a Business Administrator.
- INITIATED_BY_ME - Returns all tasks where the user calling the function is the Initiator.
- SINGLE_TASK - Returns the task identified by the P_TASK_ID input parameter.

This function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_HUMAN_TASK.GET_TASKS (
    p_context          IN VARCHAR2 DEFAULT apex_approval.c_context_my_tasks,
    p_user             IN VARCHAR2 DEFAULT apex_application.g_user,
    p_task_id          IN NUMBER   DEFAULT NULL,
    p_application_id  IN NUMBER   DEFAULT NULL,
    p_show_expired_tasks IN VARCHAR2 DEFAULT 'N' )
RETURN apex_t_approval_tasks pipelined;
```

Parameters

Parameter	Description
p_context	The list context. Default is MY_TASKS.

Parameter	Description
p_user	The user to check for. Default is logged-in user. Requires p_context set to MY_TASKS, ADMIN_TASKS or INITIATED_BY_ME.
p_task_id	Filter for a task ID instead of a user. Default is null. Requires p_context set to SINGLE_TASK.
p_application_id	Filter for an application. Default is null (all applications).
p_show_expired_tasks	If set to Y the tasks returned include tasks which are in Expired state.

Returns

A table of tasks (type apex_t_approval_tasks) containing the following columns:

- actual_owner varchar2(255)
- actual_owner_lower varchar2(255)
- app_id number
- badge_css_classes varchar2(255)
- badge_text varchar2(255)
- created_ago varchar2(255)
- created_ago_hours number
- created_by varchar2(255)
- created_on timestamp with time zone
- details_app_id number
- details_app_name varchar2(255)
- details_link_target varchar2(4000)
- due_code varchar2(32)
- due_in varchar2(255)
- due_in_hours number
- due_on timestamp with time zone
- initiator varchar2(255)
- initiator_can_complete varchar2(1)
- initiator_lower varchar2(255)
- is_completed varchar2(1)
- last_updated_by varchar2(255)
- last_updated_on timestamp with time zone
- outcome varchar2(255)
- outcome_code varchar2(32)
- priority number(1)
- priority_level varchar2(255)
- state varchar2(255)

- state_code varchar2(32)
- subject varchar2(1000)
- task_def_id number
- task_def_name varchar2(255)
- task_def_static_id varchar2(255)
- task_id number
- task_type varchar2(8)

Example

```
select * from table ( apex_human_task.get_tasks ( p_context => 'MY_TASKS',
p_show_expired_tasks => 'Y' ) )
```

31.21 HANDLE_TASK_DEADLINES Procedure

This procedure handles Task Deadlines for all Tasks in the current Workspace. A background Job performs this work every hour.

Use this API for testing of Task Expiration Policies and "Before Expire" and "Expire" Task Actions.

Syntax

```
APEX_HUMAN_TASK.HANDLE_TASK_DEADLINES
```

Parameters

Parameter	Description
none	none

Example

```
BEGIN
    apex_human_task.handle_task_deadlines;
END;
```

31.22 HAS_TASK_PARAM_CHANGED Function

This function checks if the value of this task parameter has been modified in the current session. Returns NULL when the parameter does not exist.

Syntax

```
APEX_HUMAN_TASK.HAS_TASK_PARAM_CHANGED (
    p_task_id           IN NUMBER,
    p_param_static_id   IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_param_static_id	The static ID of the parameter.

Example

```
BEGIN
    return apex_human_task.has_task_param_changed(
        p_task_id          => 1234,
        p_param_static_id => 'REQ_AMOUNT'
    );
END;
```

31.23 IS_ALLOWED Function

This function checks whether the given user is permitted to perform a certain operation on a Task.

Syntax

```
APEX_HUMAN_TASK.IS_ALLOWED (
    p_task_id           IN NUMBER,
    p_operation         IN wwv_flow_approval_api.t_task_operation,
    p_user              IN VARCHAR2 DEFAULT wwv_flow_security.g_user,
    p_new_participant  IN VARCHAR2 DEFAULT NULL )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_operation	The operation to check (see constants c_task_op_###).
p_user	The user to check for. Default is logged in user.
p_new_participant	(Optional) The new assignee in case of Delegate operation.

Returns

TRUE if the user given by p_user is permitted to perform the operation given by p_operation, FALSE otherwise.

Example

```
DECLARE
    l_is_allowed boolean;
BEGIN
    l_is_allowed := apex_human_task.is_allowed(
        p_task_id      => 1234,
        p_operation    => apex_human_task.c_task_op_delegate
```

```

        p_user          => 'STIGER',
        p_new_participant => 'SMOON'
    );
    IF l_is_allowed THEN
        dbms_output.put_line('STIGER is allowed to delegate the task to
SMOON for task 1234');
    END IF;
END;

```

31.24 IS_BUSINESS_ADMIN Function

This function checks whether the given user is a business administrator for at least one task definition.

Syntax

```
APEX_HUMAN_TASK.IS_BUSINESS_ADMIN (
    p_user          IN VARCHAR2 DEFAULT www_flow_security.g_user,
    p_application_id IN NUMBER   DEFAULT NULL )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_user	The user to check for. Default is logged-in user.
p_application_id	The application to check for. Default behavior checks against all applications in the workspace.

Returns

TRUE if the user given by p_user is at least in one task definition configured as participant type BUSINESS_ADMIN, FALSE otherwise.

Example

```

DECLARE
    l_is_business_admin boolean;
BEGIN
    l_is_business_admin := apex_human_task.is_business_admin(
        p_user => 'STIGER'
    );
    IF l_is_business_admin THEN
        dbms_output.put_line('STIGER is a Business Administrator');
    END IF;
END;

```

31.25 IS_OF_PARTICIPANT_TYPE Function

This function checks whether the given user is of a certain participant type for a Task.

Syntax

```
APEX_HUMAN_TASK.IS_OF_PARTICIPANT_TYPE (
    p_task_id           IN NUMBER,
    p_participant_type IN t_task_participant_type
                           DEFAULT c_task_potential_owner,
    p_user              IN VARCHAR2
                           DEFAULT wvv_flow_security.g_user)
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_participant_type	The participant type. Can be set to POTENTIAL_OWNER (default) or BUSINESS_ADMIN.
p_user	The user to check for. Default is logged-in user.

Returns

TRUE if the user given by p_user is a participant of given participant type for a given task, FALSE otherwise.

Example

```
DECLARE
    l_is_potential_owner boolean;
BEGIN
    l_is_potential_owner := apex_human_task.is_of_participant_type(
        p_task_id          => 1234,
        p_participant_type => apex_human_task.c_task_potential_owner,
        p_user              => 'STIGER'
    );
    IF l_is_potential_owner THEN
        dbms_output.put_line('STIGER is a potential owner for task 1234');
    END IF;
END;
```

31.26 REJECT_TASK Procedure

This procedure rejects the task. Only a potential owner or the actual owner of the task can invoke this procedure.

Moves the state of the Task to Completed and sets the outcome of the Task to Rejected. This is a convenience procedure and equivalent to calling complete_task with outcome apex_human_task.c_task_outcome_rejected.

Syntax

```
APEX_HUMAN_TASK.REJECT_TASK (
    p_task_id           IN NUMBER,
    p_autoclaim        IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_autoclaim	If Task is in state UNASSIGNED then claim the task implicitly.

State Handling

Pre-State: ASSIGNED | UNASSIGNED (p_autoclaim=true)

Post-State: COMPLETED

Example

```
BEGIN
    apex_human_task.reject_task(
        p_task_id => 1234
    );
END;
```

31.27 RELEASE_TASK Procedure

This procedure releases an Assigned task from its current owner and sets the task to Unassigned state. Only the current owner of the task can invoke this procedure.

Syntax

```
APEX_APPROVAL.RELEASE_TASK (
    p_task_id           IN NUMBER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.

State Handling

Pre-State: ASSIGNED

Post-State: UNASSIGNED

Example

```
BEGIN
    apex_human_task.release_task(
```

```

        p_task_id          => 1234
    );
END;

```

31.28 REMOVE_POTENTIAL_OWNER Procedure

This procedure removes a potential owner of a task. If the user to be removed is *not* an existing potential owner, the API raises an exception.

Only a Business Administrator for the task can run this procedure.

Syntax

```
APEX_HUMAN_TASK.REMOVE_POTENTIAL_OWNER (
    p_task_id           IN NUMBER,
    p_potential_owner   IN VARCHAR2 );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_potential_owner	The potential owner.

Example

The following example removes user "STIGER" as potential owner for Task ID 1234.

```
BEGIN
    apex_human_task.remove_potential_owner(
        p_task_id      => 1234,
        p_potential_owner => 'STIGER'
    );
END;
```

31.29 RENEW_TASK Function

This function reactivates Expired or Errored Tasks. Tasks that have been transitioned to state EXPIRED or ERRORED can be renewed by a Business Administrator.

When a Business Administrator renews a Task, a new Task is created with given the information from the given Task ID. The renewed task is associated with the Expired/Errored Task so that users can review the origin of the Task. This function returns the ID of the renewed task.

Syntax

```
APEX_HUMAN_TASK.RENEW_TASK (
    p_task_id       IN NUMBER,
    p_priority      IN INTEGER  DEFAULT NULL,
    p_due_date      IN timestamp with time zone )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_priority	The priority of the renewed Task.
p_due_date	The due date for the renewed Task.

Returns

This function returns the ID of the renewed task.

Example

```
BEGIN
    apex_human_task.renew_task(
        p_task_id          => 1234,
        p_priority         => apex_human_task.c_task_priority_high,
        p_due_date         => sysdate + 10
    );
END;
```

31.30 REQUEST_MORE_INFORMATION Procedure

This procedure requests more information for a task. The owner of a task can request additional information regarding a Task from the initiator. The task then moves to the Information Requested state and can be acted on by the owner only after the initiator submits the requested information.

Syntax

```
APEX_HUMAN_TASK.REQUEST_MORE_INFORMATION (
    p_task_id           IN NUMBER,
    p_text              IN VARCHAR2 )
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_text	Text describing the information requested.

Example

```
BEGIN
    apex_human_task.request_more_information(
        p_task_id => 1234,
        p_text     => 'Please provide the flight PNR for your travel'
    );
END;
```

31.31 SET_INITIATOR_CAN_COMPLETE Procedure

This procedure updates the `initiator_can_complete` attribute of a task. The task can **not** be COMPLETED or ERRORED. Only a user who is a business administrator for the task can invoke this procedure.

Syntax

```
APEX_HUMAN_TASK.SET_INITIATOR_CAN_COMPLETE (
    p_task_id           IN NUMBER,
    p_initiator_can_complete IN BOOLEAN )
```

Parameters

Parameter	Description
<code>p_task_id</code>	The task ID.
<code>p_initiator_can_complete</code>	TRUE if the initiator is permitted to also approve or reject the task. Otherwise, FALSE.

Example

```
BEGIN
    apex_human_task.set_initiator_can_complete(
        p_task_id      => 1234,
        p_initiator_can_complete => true
    );
END;
```

31.32 SET_TASK_DUE Procedure

This procedure sets the due date of a task and can be invoked by the Business Administrator to update the due date of the task.

This API cannot be invoked for a task that is Expired, Errorred, Completed or Canceled.

The due date needs to be in the future, otherwise an exception is thrown when invoking this API.

Syntax

```
APEX_HUMAN_TASK.SET_TASK_DUE (
    p_task_id           IN NUMBER,
    p_due_date          IN timestamp with time zone )
```

Parameters

Parameter	Description
<code>p_task_id</code>	The Task ID.
<code>p_due_date</code>	The new due date of the Task.

Example

```

BEGIN
    apex_human_task.set_task_due(
        p_task_id  => 1234,
        p_due_date => sysdate+20
    );
END;

```

31.33 SET_TASK_PARAMETER_VALUES Procedure

This procedure updates the values of the parameter(s) of this task. This procedure only updates the parameters that are marked as "updatable" in the task definition.

Only a Business Administrator or the owner of the task can run this procedure.

Syntax

```

APEX_HUMAN_TASK.SET_TASK_PARAMETER_VALUES (
    p_task_id          IN NUMBER,
    p_parameters       IN t_task_parameters,
    p_raise_error      IN BOOLEAN DEFAULT TRUE );

```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_parameters	The list of changed parameters.
p_raise_error	Default TRUE. When TRUE, the API raises an exception and cancels updates to the parameters. If FALSE, the API ignores raised exceptions if the list contains one or more incorrect parameter static IDs or parameters that are not marked as updatable in the Task Definition. The API updates the rest of the parameters.

Example

```

BEGIN
    apex_human_task.set_task_parameter_values(
        p_task_id      => 1234,
        p_parameters   => apex_human_task.t_task_parameters(
            1 => apex_human_task.t_task_parameter(static_id => 'REQ_DATE',
                                                    string_value =>
                                                    sysdate+10),
            3 => apex_human_task.t_task_parameter(static_id => 'REQ_AMOUNT',
                                                    string_value =>
                                                    l_req_amount));
END;

```

31.34 SET_TASK_PRIORITY Procedure

This procedure sets the priority of a task.

This procedure updates the priority of a task. The task can not be COMPLETED or ERRORED. Only a user who is either a Business Administrator for the task or is the initiator of the task can invoke this procedure.

Syntax

```
APEX_HUMAN_TASK.SET_TASK_PRIORITY (
    p_task_id           IN NUMBER,
    p_priority          IN INTEGER );
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_priority	The task priority (between 1 and 5, 1 being the highest).

Example

```
BEGIN
    apex_human_task.set_task_priority(
        p_task_id  => 1234,
        p_priority  => apex_human_task.c_task_priority_highest
    );
END;
```

31.35 SUBMIT_INFORMATION Procedure

This procedure submits information for a task. The initiator of a task can submit additional information regarding a Task for which information has been requested. For example, a travel approver might need airline details from the initiator. The initiator can submit this information to the travel approver using this API.

Syntax

```
APEX_HUMAN_TASK.SUBMIT_INFORMATION (
    p_task_id           IN NUMBER,
    p_text              IN VARCHAR2 )
```

Parameters

Parameter	Description
p_task_id	The Task ID.
p_text	Text containing the information submitted.

Example

```
BEGIN
    apex_human_task.submit_information(
        p_task_id => 1234,
        p_text      => 'The flight PNR is PN1234'
    );
END;
```

APEX_INSTANCE_ADMIN

The APEX_INSTANCE_ADMIN package provides utilities for managing an Oracle APEX runtime environment.

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 or SYSTEM database users and any database user granted the role APEX_ADMINISTRATOR_ROLE.

- Available Parameter Values
- ADD_AUTO_PROV_RESTRICTIONS Procedure
- ADD_SCHEMA Procedure
- ADD_WEB_ENTRY_POINT Procedure
- ADD_WORKSPACE Procedure
- CREATE_CLOUD_CREDENTIAL Procedure
- CREATE_OR_UPDATE_ADMIN_USER Procedure
- CREATE_SCHEMA_EXCEPTION Procedure
- DB_SIGNATURE Function
- DROP_CLOUD_CREDENTIAL Procedure
- FREE_WORKSPACE_APP_IDS Procedure
- GET_PARAMETER Function
- GET_SCHEMAS Function
- GET_WORKSPACE_PARAMETER Procedure
- GRANT_EXTENSION_WORKSPACE Procedure
- IS_DB_SIGNATURE_VALID Function
- REMOVE_APPLICATION Procedure
- REMOVE_AUTO_PROV_RESTRICTIONS Procedure
- REMOVE_SAVED_REPORT Procedure
- REMOVE_SAVED_REPORTS Procedure
- REMOVE_SCHEMA Procedure
- REMOVE_SCHEMA_EXCEPTION Procedure
- REMOVE_SCHEMA_EXCEPTIONS Procedure
- REMOVE_SUBSCRIPTION Procedure
- REMOVE_WEB_ENTRY_POINT Procedure
- REMOVE_WORKSPACE Procedure
- REMOVE_WORKSPACE_EXCEPTIONS Procedure

- [RESERVE_WORKSPACE_APP_IDS Procedure](#)
- [RESTRICT_SCHEMA Procedure](#)
- [REVOKE_EXTENSION_WORKSPACE Procedure](#)
- [SET_LOG_SWITCH_INTERVAL Procedure](#)
- [SET_PARAMETER Procedure](#)
- [SET_WORKSPACE_CONSUMER_GROUP Procedure](#)
- [SET_WORKSPACE_PARAMETER Procedure](#)
- [TRUNCATE_LOG Procedure](#)
- [UNLOCK_USER Procedure](#)
- [UNRESTRICT_SCHEMA Procedure](#)
- [VALIDATE_EMAIL_CONFIG Procedure](#)

32.1 Available Parameter Values

The following table lists all the available parameter values you can set within the `APEX_INSTANCE_ADMIN` package, including parameters for email, wallet, and reporting printing.

You can query `APEX_INSTANCE_PARAMETERS` dictionary view to determine the current values of these parameters unless the parameter contains a password.

Parameter Name	Description
<code>ACCOUNT_LIFETIME_DAYS</code>	The maximum number of days an end-user account password may be used before the account is expired.
<code>ADMIN_DIGEST_DEFAULT_REPORTING_PERIOD</code>	Default reporting period in days for APEX Administrator Digest.
<code>ADMIN_DIGEST_MAX_REPORTING_PERIOD</code>	Maximum reporting period in days for APEX Administrator Digest. Older data is removed from the metrics tables.
<code>ALLOW_DB_MONITOR</code>	If set to N (default), database monitoring within SQL Workshop is disabled. If Y, it is enabled.
<code>ALLOW_HASH_FUNCTIONS</code>	Comma-separated list of supported hash algorithms (default is SH256, SH384, SH512). SH1 is also supported by default in Oracle Database 11g.
<code>ALLOW_HOSTING_EXTENSIONS</code>	Determines whether a workspace can host extensions and publish links to other workspaces. Possible values are: <ul style="list-style-type: none">• N - No (default). Workspace cannot host and publish extensions.• Y - Yes, Manual Subscription. The workspace can host and publish extensions. Other workspaces have to subscribe and grant read access manually.• A - Yes, Automatic Subscription. The workspace can host extensions and extension menu entries are automatically published to all other workspaces, without these having to subscribe and grant read access manually.
<code>ALLOW_HOSTNAMES</code>	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.
<code>ALLOW_PUBLIC_FILE_UPLOAD</code>	If set to Y, enables file uploads without user authentication. If set to N, the default, they are disabled.

Parameter Name	Description
ALLOW_RAS	This parameter is only supported if running Oracle Database 12c. If Y, enables Real Application Security support for applications. If set to N (default), Real Application Security cannot be used.
APEX_BUILDER_AUTHENTICATION	Controls the authentication scheme for Oracle APEX Administration Services and the development environment. Valid parameter values include: <ul style="list-style-type: none">• APEX - Oracle APEX workspace accounts authentication (default)• DB - Database accounts authentication• HEADER - HTTP header variable based authentication• SSO - Oracle Application Server Single Sign-On authentication (OracleAS PL/SQL SSO SDK)• LDAP - LDAP authentication• SAML - SAML Sign-In authentication• SOCIAL - Social Sign-In authentication
APEX_REST_PATH_PREFIX	Controls the URI path prefix used to access built-in RESTful Services exposed by APEX. For example, built-in RESTful Service for referencing static application files using #APP_IMAGES# token. If the default prefix (r) conflicts with RESTful Services defined by users, adjust this preference to avoid the conflict.
APPLICATION_ACTIVITY_LOGGING	Controls setting of application activity log for entire instance. Options are: <ul style="list-style-type: none">• A (Always)• N (Never)• U (Use application settings)
APPLICATION_ID_MAX	The largest possible ID for a websheet or database application.
APPLICATION_ID_MIN	The smallest possible ID for a websheet or database application.
AUTHENTICATION_SUBSTITUTIONS	Enables you to specify which substitutions are visible to all authentication schemes in the APEX instance. The parameter value is a JSON format with a flat structure. Substitutions can be used for all authentication scheme attributes with #SUBSTITUTION_NAME#. For example:
	<pre>{ "SUBST_NAME": "SOME_VALUE", "SUBST_NAME_2": "VALUE_2" }</pre>
AUTOEXTEND_TABLESPACES	If set to Y (default), provisioned tablespaces are autoextended up to a maximum size. If set to N, tablespaces are not autoextended.
BIGFILE_TABLESPACES_ENABLED	If set to Y, the tablespaces provisioned through APEX are created as bigfile tablespaces. If set to N, the tablespaces are created as smallfile tablespaces.
CHECK_FOR_UPDATES	If set to N, the check for APEX and Oracle REST Data Services product updates is disabled for the entire instance, regardless of preferences specified by individual developers. The default is Y.

Parameter Name	Description
CHECKSUM_HASH_FUNCTION	Defines the algorithm that is used to create one way hashes for URL checksums. Valid values are MD5 (deprecated), SH1 (SHA-1), SH256 (SHA-2, 256 bit), SH384 (SHA-2, 384 bit), SH512 (SHA-2, 512 bit) and n. The SHA-2 algorithms are only available on Oracle Database 12g and later. A null value evaluates to the most secure algorithm available and is the default.
CLONE_SESSION_ENABLED	If set to Y, the default, users can create multiple sessions in the browser.
CONTENT_CACHE_MAX_FILE_SIZE	The individual file entry size limit for the content cache, per workspace.
CONTENT_CACHE_SIZE_TARGET	The target size for the content cache, per workspace.
DB_SIGNATURE	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.
DEBUG_MESSAGE_PAGE_VIEW_LIMIT	Maximum number of debug messages for a single page view. Default is 50000.
DELETE_UPLOADED_FILES_AFTER_DAYS	Uploaded files like application export files, websheet export files, spreadsheet data load files are automatically deleted after this number of days. Default is 14.
DISABLE_APPS_LOGIN	If set to N (default), the login to customer-created apps is enabled. If set to Y, the login to all customer-created apps is disabled. If set to a comma-separated list of application IDs, only the login to those applications is disabled.
DISABLE_ADMIN_LOGIN	If set to Y, Oracle APEX administration services are disabled. If set to N (default), they are not disabled.
DISABLE_WORKSPACE_LOGIN	If set to Y, the workspace login is disabled. If set to N, the default, the login is not disabled.
DISABLE_WS_PROV	If set to Y, the workspace creation is disabled for requests sent out by using e-mail notification. If set to N, the default, they are not disabled.
DOCGEN_CREDENTIAL	The cloud credential name to use for Oracle Object Storage bucket management and use of the Oracle Document Generator Pre-built function.
DOCGEN_FUNCTION_OCID	Specifies the Oracle Cloud Identity (OCID) of the Oracle Document Generator Pre-Built function.
DOCGEN_INVOKE_ENDPOINT	Specifies the base URL endpoint of the Oracle Document Generator Pre-Built function.
DOCGEN_OS_BUCKET_COMPARTMENT_OCID	Specifies the Oracle Cloud Identifier (OCID) of the compartment where the Oracle Document Generator Pre-built Function is located.
DOCGEN_OS_ENDPOINT	Specifies the base URL endpoint for the Oracle Object Storage bucket.
DOCGEN_OS_NAMESPACE	The Object Storage namespace serves as the top-level container for all buckets and objects.
EMAIL_ATTACHMENT_MAX_SIZE_MB	Specifies the maximum size in megabytes of a single email attachment sent using APEX_MAIL or the Send E-Mail process.
EMAIL_IMAGES_URL	Specifies the full URL to the images directory of APEX instance, including the trailing slash after the images directory. For example: <code>http://your_server/i/</code>
	This setting is used for APEX system-generated emails.

Parameter Name	Description
EMAIL_INSTANCE_URL	Specifies the URL to APEX instance, including the trailing slash after the Database Access Descriptor. For example: <code>http://your_server/pls/apex/</code> This setting is used for APEX system-generated emails.
ENABLE_LEGACY_WEB_ENTRY_POINTS	If set to Y (default is N), procedures used in older APEX versions can be called in the URL (such as <code>HTMLDB_UTIL.%</code>).
ENABLE_TRANSACTIONAL_SQL	If set to Y, transactional SQL commands are enabled on this instance. If set to N, the default, they are not enabled.
ENCRYPTED_TABLESPACES_ENABLED	If set to Y, the tablespaces provisioned through APEX are created as encrypted tablespaces. If set to N, the tablespaces are not encrypted.
ENV_BANNER_COLOR	Defines the color class name for the environment banner color. Use accent-1, accent-2, accent-3, (and so on). Maximum of 16 color classes.
ENV_BANNER_ENABLE	Default N. If set to N, the default, the banner does not display. If set to Y, the environment banner displays in the APEX development environment to visually flag the environment.
ENV_BANNER_LABEL	Defines the label for the environment banner.
ENV_BANNER_POS	Defines the display position for the environment banner. Options: LEFT or TOP.
EXPIRE_FND_USER_ACCOUNTS	If set to Y, expiration of APEX accounts is enabled. If set to N, they are not enabled.
HEADER_AUTH_CALLBACK	Callback procedure name for HTTP header based authentication, defaults to <code>apex_authentication.callback</code> .
HTTP_ERROR_STATUS_ON_ERROR_PAGE_ENABLED	Used in conjunction with the <code>APEX_INSTANCE_ADMIN.SET_PARAMETER</code> procedure. If set to N, the default, APEX 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 APEX engine encounters an unhandled error.
HTTP_RESPONSE_HEADERS	List of http response headers, separated by newline (<code>chr(10)</code>). APEX writes these headers on each request, before rendering the page. The substitution string <code>#CDN#</code> within the headers is replaced with the content delivery networks that are known to APEX.
HTTP_STS_MAX_AGE	<code>REQUIRE_HTTPS</code> must be set to A for this parameter to be relevant. APEX emits a Strict-Transport-Security header, with <code>max-age=<value></code> , on HTTPS requests if <code>HTTP_STS_MAX_AGE</code> has a value greater than 0. If the request protocol is HTTP, instead of processing the request, APEX redirects to a HTTPS URL.
HTTP_TRUSTED_ORIGINS	List of remote HTTP origins that can access resources, separated by newline. Set this parameter in combination with the ORDS parameter <code>security.externalSessionTrustedOrigins</code> .

Parameter Name	Description
IGNORED_FRIENDLY_URL_PARAMETERS	Comma-separated list of parameter names which are ignored when parsing friendly URLs. Default: utm_campaign,utm_source,utm_medium,utm_term,utm_content
INBOUND_PROXIES	Comma-separated list of IP addresses for proxy servers through which requests come in.
INSTANCE_DBMS_CREDENTIAL_ENABLED	If set to Y, database credentials that are accessible to the APEX engine (APEX_NNNNN schema), can be used in all workspaces on this instance.
INSTANCE_NO_PROXY_DOMAINS	Comma-separated list of domain names for which the instance proxy is not to be used.
INSTANCE_PROXY	The proxy server for all outbound HTTP(s) traffic. If INSTANCE_PROXY is set, it overrides any application specific proxy server definition.
INSTANCE_TABLESPACE	If specified, the tablespace to use for the database user for all new workspaces.
KEEP_SESSIONS_ON_UPGRADE	This flag affects application upgrades. If set to N, the default, delete sessions associated with the application. If set to Y, leave sessions unaffected.
LOGIN_MESSAGE	The text to be displayed on the login page. This text can include HTML.
LOGIN_THROTTLE_DELAY	The flag which determines the time increase in seconds after failed logins.
LOGIN_THROTTLE_METHODS	The methods to count failed logins. Colon-separated list of USERNAME_IP, USERNAME, IP.
MAX_APPLICATION_BACKUPS	The maximum number of backups kept for each application. Default is 25. Maximum is 30. Zero (0) disables automated backups.
MAX_DATA_EXPORT_IMAGES	The maximum number of unique images to be included in a data export / report download.
MAX_LOGIN_FAILURES	Maximum login failures permitted.
MAX_MAIL_QUEUE_ROWS	Defines the number of email messages that are processed from the queue per workspace during each invocation of the ORACLE_APEX_MAIL_QUEUE scheduler job.
MAX_SESSION_IDLE_SEC	The number of seconds an internal application may be idle.
MAX_SESSION_LENGTH_SEC	The number of seconds an internal application session may exist.
MAX_WEBSERVICE_REQUESTS	The maximum number of outbound web service requests permitted for each workspace in a rolling 24-hour period. Default is 1000.
PASSWORD_ALPHA_CHARACTERS	The alphabetic characters used for password complexity rules. Default list of alphabetic characters include the following: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Parameter Name	Description
PASSWORD_HASH_FUNCTION	Defines the algorithm that is used to create one way hashes for workspace user passwords. Valid values are MD5 (deprecated), SH1 (SHA-1), SH256 (SHA-2, 256 bit), SH384 (SHA-2, 384 bit), SH512 (SHA-2, 512 bit) and null. The SHA-2 algorithms are only available on Oracle Database Release 12g and later. A null value evaluates to the most secure algorithm available and is the default.
PASSWORD_HASH_ITERATIONS	Defines the number of iterations for the PASSWORD_HASH_FUNCTION (default 10000).
PASSWORD_HISTORY_DAYS	Defines the number of days a previously used password cannot be used again as a new password by the same user.
PASSWORD_NOT_LIKE_USERNAME	If Y (the default is N), prevent workspace administrator, developer, and end user account passwords from containing the username.
PASSWORD_NOT_LIKE_WORDS	Enter words, separated by colons, that workspace administrator, developer, and end user account passwords must not contain. These words may not appear in the password in any combination of upper- or lowercase.
PASSWORD_NOT_LIKE_WS_NAME	Set to Y to prevent workspace administrator, developer, and end user account passwords from containing the workspace name.
PASSWORD_ONE_ALPHA	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one alphabetic character as specified in PASSWORD_ALPHA_CHARACTERS.
PASSWORD_ONE_LOWER_CASE	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one lowercase alphabetic character.
PASSWORD_ONE_NUMERIC	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one Arabic numeric character (0-9).
PASSWORD_ONE_PUNCTUATION	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one punctuation character as specified in PASSWORD_PUNCTUATION_CHARACTERS.
PASSWORD_ONE_UPPER_CASE	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one uppercase alphabetic character.
PASSWORD_PUNCTUATION_CHARACTERS	The punctuation characters used for password complexity rules. Default list of punctuation characters include the following: !"#\$%&()``*+,.-/:;<=>?_
PATH_PREFIX	The unique URI path prefix used to access RESTful Services in a workspace. The default path prefix value is the name of the workspace.
PLSQL_EDITING	If set to Y, the default, the SQL Workshop Object Browser is enabled to permit users to edit and compile PL/SQL. If set to N, users are not permitted.

Parameter Name	Description
PRINT_BIB_LICENSED	Specify either standard support or advanced support. Advanced support requires an Oracle BI Publisher license. Valid values include: <ul style="list-style-type: none"> • STANDARD - requires Apache FOP. • DOCUMENT_GENERATOR - requires Oracle Document Generator Pre-built Function. • ADVANCED - requires Oracle BI Publisher. • AOP - requires APEX Office Print. • NONE - native APEX printing.
PRINT_SVR_HOST	Specifies the host address of the print server converting engine, for example, localhost. Enter the appropriate host address if the print server is installed at another location.
PRINT_SVR_PORT	Defines the port of the print server engine, for example 8888. Value must be a positive integer.
PRINT_SVR_PROTOCOL	Valid values include: <ul style="list-style-type: none"> • http • https
PRINT_SVR_SCRIPT	Defines the script that is the print server engine, for example: /xmlpserver/convert
QOS_MAX_SESSION_KILL_TIMEOUT	Number of seconds that an active old session can live, when QOS_MAX_SESSION_REQUESTS has been reached. The oldest database session with LAST_CALL_ET greater than QOS_MAX_SESSION_KILL_TIMEOUT is canceled.
QOS_MAX_SESSION_REQUESTS	Number of permitted concurrent requests to one session associated with this workspace.
QOS_MAX_WORKSPACE_REQUESTS	Number of permitted concurrent requests to sessions in this workspace.
REJOIN_EXISTING_SESSIONS	If P (default), session rejoining is supported for non-authenticated users and public pages. If Y, session rejoining is also supported for authenticated users and protected pages. If N, session rejoining is disabled for the whole instance. Session rejoining must be set to enabled at application or page level. A more restrictive setting at instance level with this instance parameter overrides application and page settings. Unconditionally enabling session rejoining has serious security implications. Attackers could take over sessions via XSS or if they have development access to a workspace.
REQ_NEW_SCHEMA	If set to Y, the option for new schema for new workspace requests is enabled. If set to N, the default, the option is disabled.

Parameter Name	Description
REQUIRE_HTTPS	If set to A, enforces HTTPS for the entire APEX instance. If I, enforces HTTPS within the APEX development and administration applications. If N, permits all applications to be used when the protocol is either HTTP or HTTPS.
	<p> Note:</p> <p>Note developers can also enforce HTTPS at the application level, by setting the Secure attribute of an application scheme's cookie.</p>
RESTFUL_SERVICES_ENABLED	If set to Y, the default, RESTful services development is enabled. If set to N, RESTful services are not enabled.
RESTRICT_DEV_HEADER	Controls access to the APEX development environment and Administration Services using an HTTP request header. Specify the name of the header, for example Public-Access. If this header exists in the request, access is blocked. Normally an external load balancer or a web server adds this header. The value of the header is ignored.
RESTRICT_APPS_HEADER	To restrict access to a specific list of applications, enter a HTTP request header name. If the header exists, logging into the application is only allowed if the application ID is contained in the comma-delimited list of applications in the header.
RESTRICT_IP_RANGE	To restrict access to the APEX development environment and Administration Services to a specific range of IP addresses, enter a comma-delimited list of IP addresses. If necessary, you can use an asterisk (*) as a wildcard, but do not include additional numeric values after wildcard characters. For example, 138.*.41.2 is not a valid value.
RESTRICT_RESPONSE_HEADERS	If Y or null (default), show HTTP 500 when a page contains unsupported HTTP response headers. These include status codes 301, 308 and 410, and cache headers for POST requests.
RM_CONSUMER_GROUP	If set, this is the resource manager consumer group to be used for all page events. A more specific group can be configured at workspace level.
SAMESITE_COOKIE	Default value of the cookie attribute "samesite."
SAML_APEX_CALLBACK_URLS	SAML authentication: Supported URLs for apex_authentication.saml_callback, separated by newlines. If set, APEX verifies that the domain in the browser is part of this list and sends its index (starting at 0) in authentication requests as AssertionConsumerServiceIndex.
SAML_APEX_CERTIFICATE	SAML authentication: The primary certificate of the APEX side.
SAML_APEX_CERTIFICATE2	(Optional) SAML authentication: The alternative certificate of the APEX side.
SAML_APEX_PRIVATE_KEY	SAML authentication: The private key of the APEX side.

Parameter Name	Description
SAML_APEX_PRIVATE_KEY2	(Optional) SAML authentication: The alternative private key of the APEX side.
SAML_ENABLED	SAML authentication: Y if workspace applications should be able to use SAML authentication.
SAML_IP_ISSUER	SAML authentication: Issuer attribute from the identity provider's metadata.
SAML_IP_SIGNING_CERTIFICATE	SAML authentication: The certificate from the identity provider's metadata.
SAML_IP_SIGNING_CERTIFICATE2	(Optional) SAML authentication: An alternative certificate from the identity provider's metadata.
SAML_NAMEID_FORMAT	SAML authentication: The NameID format that APEX expects. Defaults to urn:oasis:names:tc:SAML:2.0:nameid-format:persistent when null.
SAML_SIGN_IN_URL	SAML authentication: The identity provider's sign in URL.
SAML_SIGN_OUT_URL	(Optional) SAML authentication: The identity provider's sign out URL.
SAML_SP_ISSUER	SAML authentication: The "issuer" attribute that APEX sends (defaults to the callback URL).
SAML_USERNAME_ATTRIBUTE	SAML authentication: Responses can contain additional attributes about the user. If set, APEX uses that attribute's value as the username (defaults to the assertion subject's NameID attribute).
SERVICE_ADMIN_PASSWORD_MIN_LENGTH	A positive integer or 0 which specifies the minimum character length for passwords for instance administrators, workspace administrators, developers, and end user APEX accounts, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_NEW_DIFFERS_BY	A positive integer or 0 which specifies the number of differences required between old and new passwords. The passwords are compared character by character, and each difference that occurs in any position counts toward the required minimum difference. This setting applies to accounts for instance administrators, workspace administrators, developers, and end user APEX accounts, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_ONE_ALPHA	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one alphabetic character as specified in PASSWORD_ALPHA_CHARACTERS , when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_ONE_NUMERIC	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one Arabic numeric character (0-9), when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_ONE_PUNCTUATION	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one punctuation character as specified in PASSWORD_PUNCTUATION_CHARACTERS , the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_ONE_LOWER_CASE	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one lowercase alphabetic character, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).

Parameter Name	Description
SERVICE_ADMIN_PASSWORD_ONE_UP_PER_CASE	Set to Y to require that workspace administrator, developer, and end user account passwords contain at least one uppercase alphabetic character, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_NOT_LIKE_USERNAME	If Y, prevent workspace administrator, developer, and end user account passwords from containing the username, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_ADMIN_PASSWORD_NOT_LIKE_WORDS	Enter words, separated by colons, that workspace administrator, developer, and end user account passwords must not contain, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD). These words may not appear in the password in any combination of upper- or lowercase.
SERVICE_ADMIN_PASSWORD_NOT_LIKE_WS_NAME	Set to Y to prevent workspace administrator, developer, and end user account passwords from containing the workspace name, when the strong password rules are enabled (see STRONG_SITE_ADMIN_PASSWORD).
SERVICE_REQUEST_FLOW	Determines default provisioning mode. Options are: <ul style="list-style-type: none"> • MANUAL - (Default) An administrator must manually create each workspace. • EMAIL - Link displayed on login page. Requests require administrator approval. • AUTO - Link displayed on login page. Requests are automatically approved.
SERVICE_REQUESTS_ENABLED	If set to Y, the default, workspace service requests for schemas, storage, and termination is enabled. If set to N, these requests are disabled.
SESSION_TIMEOUT_WARNING_SEC	The number of seconds before session timeout that a warning displays for internal applications.
SMTP_FROM	Defines the "From" address for administrative tasks that generate email, such as approving a provision request or resetting a password. Enter a valid email address, for example:
	admin@example.com
SMTP_HOST_ADDRESS	Defines the server address of the SMTP server. If you are using another server as an SMTP relay, change this parameter to that server's address. Default setting: localhost
SMTP_HOST_PORT	Defines the port the SMTP server listens to for mail requests. Default setting: 25
SMTP_PASSWORD	Defines the password APEX takes to authenticate itself against the SMTP server, with the parameter SMTP_USERNAME .

Parameter Name	Description
SMTP_TLS_MODE	Defines whether APEX 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, APEX sends the SMTP commands EHLO <SMTP_HOST_ADDRESS> and STARTTLS before encrypting the connection.
SMTP_USERNAME	Defines the username APEX takes to authenticate itself against the SMTP server (default is null). Starting with database version 11.2.0.2, APEX 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.
SOCIAL_AUTH_CALLBACK	Callback procedure name for Social Sign-In, defaults to apex_authentication.callback.
SQL_SCRIPT_MAX_OUTPUT_SIZE	Sets the maximum size for an individual script result. Default is 200000.
SSO_LOGOUT_URL	Defines the URL APEX redirects to in order to trigger a logout from the Single Sign-On server. APEX automatically appends ?p_done_url=...login url... Example: https://login.example.com/pls/orasso/orasso.wwsso_app_admin.ls_logout
STRONG_SITE_ADMIN_PASSWORD	If set to Y, the default, the apex_admin password must conform to the default set of strong complexity rules. If set to N, the password is not required to follow the strong complexity rules.
SYSTEM_DEBUG_LEVEL	Defines a default debug level for all incoming requests (null, 1-9) The SQLcl script utilities/debug/d0.sql can be used to switch between NULL (disabled) and level 9.
SYSTEM_HELP_URL	Location of the help and documentation accessed from the Help link within the development environment. Default is http://apex.oracle.com/doc41
SYSTEM_MESSAGE	The text to be displayed on the development environment home page. This text can include HTML.
TRACE_HEADER_NAME	This parameter contains a HTTP request header name and defaults to ECID-CONTEXT. The name must be in upper case. APEX writes the HTTP header value to the activity log's ECID column.
TRACING_ENABLED	If set to Y (the default), an application with Debug enabled can also generate server side db trace files using &p_trace=YES on the URL. If set to N, the request to create a trace file is ignored.
UPGRADE_DATE	This read-only parameter contains the date when the next scheduled APEX upgrade will automatically apply to your database or NULL when no upgrade is scheduled. The date follows the ISO 8601 format in the UTC time zone. This parameter only applies to APEX on Autonomous Database.

Parameter Name	Description
UPGRADE_DEFERRED	When N (default), future APEX upgrades are scheduled within the default upgrade window. When Y, future APEX upgrades are scheduled within the extended upgrade window. If the next APEX upgrade is already scheduled in this database, changing this parameter also reschedules this upgrade (for example, from the default upgrade window to the extended upgrade window). This parameter only applies to APEX on Oracle Autonomous Database.
UPGRADE_STATUS	This parameter contains the status of the next scheduled APEX upgrade if available. The possible values are: UP-TO-DATE, SCHEDULED, RUNNING. Set this parameter to RUN to initiate the upgrade. This parameter only applies to APEX on Autonomous Database.
UPGRADE_VERSION	This read-only parameter contains the APEX version to be installed with the next scheduled upgrade or NULL when no upgrade is scheduled. This parameter only applies to APEX on Autonomous Database.
USERNAME_VALIDATION	The case-sensitive regular expression used to validate a username when creating or modifying developer and workspace administrator accounts. Default is * (asterisk) (validation is disabled).
WALLET_PATH	The path to the wallet on the file system, for example: file:/home/<username>/wallets
WALLET_PWD	The password associated with the wallet. Use an empty/null value for auto-login wallets.
WEBSERVICE_LOGGING	Controls instance wide setting of web service activity log. A, N, or U (Always, Never, Use workspace settings).
WORKSPACE_EMAIL_MAXIMUM	Sets the maximum number of emails that can be sent by using APEX_MAIL per workspace in a 24-hour period. Default is 1000.
WORKSPACE_FREE_SPACE_LIMIT	Sets percentage limit for free space in a workspace. If available space is lower than the value set here, a report lists them for the APEX Administrator Digest.
WORKSPACE_MAX_FILE_BYTES	The maximum number of bytes for uploaded files for a workspace. A setting at the workspace-level overrides the instance-level setting.
WORKSPACE_MAX_OUTPUT_SIZE	The maximum space allocated for script results. Default is 2000000
WORKSPACE_NAME_USER_COOKIE	If set to Y or null (the default), APEX sends persistent cookies for workspace name and username during login, as well as for language selection. If N, the cookies are not sent.
WORKSPACE_PROVISION_DEMO_OBJECTS	If set to Y (default), demonstration applications and database objects are created in new workspaces. If set to N, they are not created in new workspaces.
WORKSPACE_TEAM_DEV_FILES_YN	If set to Y, the default, new workspaces enable file uploads into Team Development. If set to N, new workspaces disable file uploads into Team Development, disabling the ability to upload feature, bug, and feedback attachments.

Parameter Name	Description
WORKSPACE_TEAM_DEV_FS_LIMIT	The maximum per upload file size of a Team Development file (feature, bug, and feedback attachments). Default value is 15728640 (15 MB). All possible options are listed below: 5 MB - 5242880 10 MB - 10485760 15 MB - 15728640 20 MB - 20971520 25 MB - 26214400
ZIP_FILE_MAX_EXPANSION_FACTOR	The maximum factor by which a compressed file can expand after decompression. Default value is 200.
ZIP_FILE_MAX_UNCOMPRESSED_SIZE_MB	The maximum size to unzip a file. Default value is 4096 MB.

 **See Also:**

- Configuring Email in a Runtime Environment in the *Oracle APEX Administration Guide*
- Configuring Wallet Information in the *Oracle APEX Administration Guide*
- Configuring Report Printing for an Instance in the *Oracle APEX Administration Guide*
- Workspace and Application Administration in the *Oracle APEX Administration Guide*

32.2 ADD_AUTO_PROV_RESTRICTIONS Procedure

This procedure adds blocking email patterns when an instance has auto-provisioning or self-provisioning enabled for workspaces.

If auto/self-provisioning is disabled, this procedure has no runtime effect.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_AUTO_PROV_RESTRICTIONS (
    p_block_email_patterns IN apex_t_varchar2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_block_email_patterns	Add one or more email patterns to be removed from the apex_auto_prov_email_patterns table.

Example

```
BEGIN
    apex_instance_admin.add_auto_prov_restrictions (
        p_block_email_patterns =>
```

```
apex_t_varchar2('%@gmail.com','%@example.com') );  
END;
```

32.3 ADD_SCHEMA Procedure

This procedure adds a schema to a workspace to schema mapping.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_SCHEMA (   
    p_workspace          IN VARCHAR2,  
    p_schema             IN VARCHAR2,  
    p_grant_apex_privileges IN VARCHAR2 DEFAULT FALSE )
```

Parameters

Parameter	Description
p_workspace	The name of the workspace to which the schema mapping is added.
p_schema	The schema to add to the schema to workspace mapping.
p_grant_apex_privileges	Grant the privileges needed by Oracle APEX to this schema. Default FALSE.

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',true );  
END;
```

32.4 ADD_WEB_ENTRY_POINT Procedure

Purpose

Add a public procedure to the list of allowed objects that can be called via the URL.

The parsing schema (such as APEX_PUBLIC_USER) must have privileges to execute the procedure. You must enable EXECUTE TO PUBLIC or the parsing schema.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_WEB_ENTRY_POINT (   
    p_name      IN VARCHAR2,  
    p_methods   IN VARCHAR2 DEFAULT 'GET' );
```

Parameters

Parameter	Description
p_name	The procedure name, prefixed by package name and schema, unless a public synonym exists.
p_methods (deprecated)	<p> Note:</p> <p>This parameter is deprecated and will be removed in a future release.</p> <p>The comma-separated HTTP request methods (such as GET or POST). Default GET.</p>

Examples

This example enables myschema.mypkg.proc to be called via GET and POST requests, such as <https://www.example.com/apex/myschema.mypkg.proc>

```
BEGIN
    apex_instance_admin.add_web_entry_point (
        p_name      => 'MYSCHHEMA.MYPKG.PROC',
        p_methods   => 'GET,POST' );
    COMMIT;
END;
```

32.5 ADD_WORKSPACE Procedure

Adds a workspace to an Oracle APEX Instance.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_WORKSPACE(
    p_workspace_id      IN NUMBER DEFAULT NULL,
    p_workspace         IN VARCHAR2,
    p_source_identifier IN VARCHAR2 DEFAULT NULL,
    p_primary_schema    IN VARCHAR2,
    p_additional_schemas IN VARCHAR2,
    p_rm_consumer_group IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_workspace_id	The ID to uniquely identify the workspace in an APEX instance. This may be left null and a new unique ID is assigned.
p_workspace	The name of the workspace to be added.
p_source_identifier	A short identifier for the workspace used when synchronizing feedback between different instances.
p_primary_schema	The primary database schema to associate with the new workspace.

Parameter	Description
p_additional_schemas	A colon delimited list of additional schemas to associate with this workspace.
p_rm_consumer_group	Resource Manager consumer group which is used when executing applications of this workspace.

Example

The following example demonstrates how to use the ADD_WORKSPACE procedure to add a new workspace named MY_WORKSPACE using the primary schema, SCOTT, along with additional schema mappings for HR and OE.

```
BEGIN
    APEX_INSTANCE_ADMIN.ADD_WORKSPACE (
        p_workspace_id      => 8675309,
        p_workspace          => 'MY_WORKSPACE',
        p_primary_schema     => 'SCOTT',
        p_additional_schemas => 'HR:OE' );
END;
```

32.6 CREATE_CLOUD_CREDENTIAL Procedure

This procedure creates a new DBMS_CLOUD OCI API Key credential. This procedure creates a credential in DBMS_CLOUD using DBMS_CLOUD.CREATE_CREDENTIAL.

Syntax

```
APEX_INSTANCE_ADMIN.CREATE_CLOUD_CREDENTIAL (
    p_credential_name      IN VARCHAR2,
    p_user_ocid            IN VARCHAR2,
    p_tenancy_ocid         IN VARCHAR2,
    p_private_key          IN VARCHAR2,
    p_fingerprint          IN VARCHAR2 )
```

Parameters

Parameter	Description
p_credential_name	Name for credential.
p_user_ocid	Oracle Cloud identifier (OCID) for the user.
p_tenancy_ocid	Oracle Cloud identifier (OCID) for the tenancy.
p_private_key	Private key.
p_fingerprint	Specifies a fingerprint.

Example

The following example creates the MY_CREDENTIAL credential.

```
BEGIN
    APEX_INSTANCE_ADMIN.CREATE_CLOUD_CREDENTIAL (
        p_credential_name      => 'MY_CREDENTIAL',
```

```

    p_user_ocid          => 'ocid1.user.oc1...',
    p_tenancy_ocid       => 'ocid1.tenancy.oc1...',
    p_private_key        => 'ABCDEFGHIJKLMNPQRSTUVWXYZ',
    p_fingerprint       =>
'12:34:56:78:90:ab:cd:ef:12:34:56:78:90:ab:cd:ef' );
END;

```

32.7 CREATE_OR_UPDATE_ADMIN_USER Procedure

This procedure creates an instance administration user account (that is, a user in the INTERNAL workspace). If the account already exists, this procedure also unlocks it and updates the account with a random password (not used when the builder authentication is Database Accounts).

This is the procedural equivalent of calling the `apxchpwd.sql` script.

Syntax

```
APEX_INSTANCE_ADMIN.CREATE_OR_UPDATE_ADMIN_USER (
    p_username    IN VARCHAR2 )
```

Parameters

Parameter	Description
p_username	The username.

Example

The following example creates or updates the user ADMIN.

```

BEGIN
    apex_instance_admin.create_or_update_admin_user (
        p_username => 'ADMIN',
        COMMIT;
END;
```

32.8 CREATE_SCHEMA_EXCEPTION Procedure

This procedure creates an exception which allows assignment of a restricted schema to a specific workspace.

Syntax

```
APEX_INSTANCE_ADMIN.CREATE_SCHEMA_EXCEPTION (
    p_schema      IN VARCHAR2,
    p_workspace   IN VARCHAR2 )
```

Parameter

Parameter	Description
p_schema	The schema.
p_workspace	The workspace.

Example

This example allows the assignment of restricted schema HR to workspace HR_WORKSPACE.

```
BEGIN
    apex_instance_admin.create_schema_exception (
        p_schema      => 'HR',
        p_workspace   => 'HR_WORKSPACE' );
    COMMIT;
END;
```

 **See Also:**

- [RESTRICT_SCHEMA Procedure](#)
- [UNRESTRICT_SCHEMA Procedure](#)
- [REMOVE_SCHEMA_EXCEPTION Procedure](#)
- [REMOVE_SCHEMA_EXCEPTIONS Procedure](#)
- [REMOVE_WORKSPACE_EXCEPTIONS Procedure](#)

32.9 DB_SIGNATURE Function

This function computes the current database signature value.

Syntax

```
APEX_INSTANCE_ADMIN.DB_SIGNATURE
    RETURN VARCHAR2;
```

Example

The following example sets the DB_SIGNATURE instance parameter to the current database signature.

```
BEGIN
    apex_instance_admin.set_parameter (
        p_parameter => 'DB_SIGNATURE',
        p_value      => apex_instance_admin.db_signature );
END;
```

 See Also:

- [IS_DB_SIGNATURE_VALID Function](#)
- [Available Parameter Values](#)

32.10 DROP_CLOUD_CREDENTIAL Procedure

This procedure Drops an existing DBMS_CLOUD OCI API Key credential. The procedure drops a credential in the internal Oracle APEX schema using `DBMS_CLOUD.DROP_CREDENTIAL`.

Syntax

```
APEX_INSTANCE_ADMIN.DROP_CLOUD_CREDENTIAL (
    p_credential_name      IN VARCHAR2 )
```

Parameters

Parameter	Description
<code>p_credential_name</code>	Name for credential.

Example

The following example drops the `MY_CREDENTIAL` credential.

```
BEGIN
    APEX_INSTANCE_ADMIN.DROP_CLOUD_CREDENTIAL (
        p_credential_name      => 'MY_CREDENTIAL' );
END;
```

32.11 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

Parameter	Description
<code>p_workspace_id</code>	The unique ID of the workspace.

Example

This example illustrates how to undo the reservation of application IDs that belong to a workspace with an ID of 1234567890.

```
begin
    apex_instance_admin.free_workspace_app_ids(1234567890);
end;
```

32.12 GET_PARAMETER Function

This 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

Parameter	Description
p_parameter	The instance parameter to be retrieved. See Available Parameter Values .

Example

The following example demonstrates how to use the `GET_PARAMETER` function to retrieve the `SMTP_HOST_ADDRESS` parameter currently defined for an APEX 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;
```

32.13 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

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

Example

The following example demonstrates how to use the GET_SCHEMA function to retrieve the underlying schemas mapped to a workspace.

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

32.14 GET_WORKSPACE_PARAMETER Procedure

Gets the workspace parameter.

Syntax

```
APEX_INSTANCE_ADMIN.GET_WORKSPACE_PARAMETER (
    p_workspace      IN VARCHAR2,
    p_parameter      IN VARCHAR2 )
```

Parameters

Parameter	Description
p_workspace	The name of the workspace from which you are getting the workspace parameter.

Parameter	Description
p_parameter	<p>The name of the parameter to get. Parameter names include:</p> <ul style="list-style-type: none"> • ACCOUNT_LIFETIME_DAYS • ALLOW_HOSTNAMES • ENV_BANNER_COLOR • ENV_BANNER_LABEL • ENV_BANNER_POS • ENV_BANNER_YN • EXPIRE_FND_USER_ACCOUNTS • MAX_LOGIN_FAILURES • MAX_SESSION_IDLE_SEC • MAX_SESSION_LENGTH_SEC • MAX_WEBSERVICE_REQUESTS • QOS_MAX_SESSION_KILL_TIMEOUT • QOS_MAX_SESSION_REQUESTS • QOS_MAX_WORKSPACE_REQUESTS • RM_CONSUMER_GROUP • WEBSERVICE_LOGGING • WORKSPACE_EMAIL_MAXIMUM • WORKSPACE_MAX_FILE_BYTES

Example

The following example prints the value of ALLOW_HOSTNAMES for the HR workspace.

```
BEGIN
    DBMS_OUTPUT.PUT_LINE (
        APEX_INSTANCE_ADMIN.GET_WORKSPACE_PARAMETER (
            p_workspace => 'HR',
            p_parameter => 'ALLOW_HOSTNAMES' ));
END;
```

32.15 GRANT_EXTENSION_WORKSPACE Procedure

This procedure grants read access for a workspace to an extension workspace. Builder extension menu links of the extension workspace appear in the grantor workspace's extension menu.

Syntax

```
APEX_INSTANCE_ADMIN.GRANT_EXTENSION_WORKSPACE (
    p_from_workspace      IN VARCHAR2,
    p_to_workspace        IN VARCHAR2,
    p_read_access         IN BOOLEAN DEFAULT FALSE,
    p_menu_label          IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_from_workspace	Name of workspace granting access.
p_to_workspace	Name of extension workspace.
p_read_access	Default FALSE.
p_menu_label	If TRUE, the extension workspace has read access to the grantor's repository views. (Optional) Overwrite the extension menu parent label. Otherwise, shows the name of the extension workspace.

Example

The following example grants extension workspace EXTENSIONS read access to workspace MY_WORKSPACE and overwrites the extension menu parent label with "Tools."

```
BEGIN
    apex_instance_admin.grant_extension_workspace(
        p_from_workspace      => 'MY_WORKSPACE',
        p_to_workspace        => 'EXTENSIONS',
        p_read_access         => true,
        p_menu_label          => 'Tools' );
END;
```

32.16 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

```
APEX_INSTANCE_ADMIN.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](#)

32.17 REMOVE_APPLICATION Procedure

Removes the application specified from the Oracle APEX instance.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_APPLICATION (
    p_application_id IN NUMBER );
```

Parameters

Parameter	Description
p_application_id	The ID of the application.

Example

The following example demonstrates how to use the REMOVE_APPLICATION procedure to remove an application with an ID of 100 from an APEX instance.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_APPLICATION(100);
END;
```

32.18 REMOVE_AUTO_PROV_RESTRICTIONS Procedure

This procedure removes blocking email patterns when an instance has auto-provisioning or self-provisioning enabled for workspaces.

If auto/self-provisioning is disabled, this procedure has no runtime effect.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_AUTO_PROV_RESTRICTIONS (
    p_block_email_patterns IN apex_t_varchar2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_block_email_patterns	Add one or more email patterns to be removed from the apex_auto_prov_email_patterns table.

Example

```
BEGIN
    apex_instance_admin.remove_auto_prov_restrictions (
        p_block_email_patterns =>
        apex_t_varchar2('%@gmail.com','%@example.com') );
END;
```

32.19 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

Parameter	Description
<code>p_application_id</code>	The ID of the application for which to remove user saved interactive report information.
<code>p_report_id</code>	The ID of the saved user interactive report to be removed.

Example

The following example demonstrates how to use the `REMOVE_SAVED_REPORT` procedure to remove user saved interactive report with the ID 123 for the application with an ID of 100.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(100,123);
END;
```

32.20 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

Parameter	Description
p_application_id	The ID of the application for which to remove user-saved interactive report information. If NULL, all user-saved interactive reports for the entire instance is removed.

Example

The following example demonstrates how to use the REMOVE_SAVED_REPORTS procedure to remove user saved interactive report information for the application with an ID of 100.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(100);
END;
```

32.21 REMOVE_SCHEMA Procedure

Removes a workspace to schema mapping.

Syntax

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

Parameters

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

Example

The following example demonstrates how to use the REMOVE_SCHEMA procedure to remove the schema named Frank from the MY_WORKSPACE workspace to schema mapping.

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

32.22 REMOVE_SCHEMA_EXCEPTION Procedure

This procedure removes an exception that allows the assignment of a restricted schema to a given workspace.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA_EXCEPTION (
    p_schema    IN VARCHAR2,
    p_workspace IN VARCHAR2 )
```

Parameter

Parameter	Description
p_schema	The schema.
p_workspace	The workspace.

Example

This example removes the exception that allows the assignment of schema `HR` to workspace `HR_WORKSPACE`.

```
BEGIN
    apex_instance_admin.remove_schema_exception (
        p_schema    => 'HR',
        p_workspace => 'HR_WORKSPACE' );
    commit;
END;
```

See Also:

- [CREATE_SCHEMA_EXCEPTION Procedure](#)
- [RESTRICT_SCHEMA Procedure](#)
- [UNRESTRICT_SCHEMA Procedure](#)
- [REMOVE_SCHEMA_EXCEPTIONS Procedure](#)
- [REMOVE_WORKSPACE_EXCEPTIONS Procedure](#)

32.23 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

Parameter	Description
p_schema	The schema.

Example

This example removes all exceptions that allow the assignment of the `HR` schema to workspaces.

```
BEGIN
    apex_instance_admin.remove_schema_exceptions (
        p_schema => 'HR' );
    COMMIT;
END;
```

See Also:

- [CREATE_SCHEMA_EXCEPTION Procedure](#)
- [RESTRICT_SCHEMA Procedure](#)
- [UNRESTRICT_SCHEMA Procedure](#)
- [REMOVE_SCHEMA_EXCEPTION Procedure](#)
- [REMOVE_WORKSPACE_EXCEPTIONS Procedure](#)

32.24 REMOVE_SUBSCRIPTION Procedure

Removes a specific interactive report subscription.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SUBSCRIPTION (
    p_subscription_id      IN NUMBER )
```

Parameters

Parameter	Description
p_subscription_id	The ID of the interactive report subscription to be removed.

Example

The following example demonstrates how to use the REMOVE_SUBSCRIPTION procedure to remove interactive report subscription with the ID 12345. Use of APEX_APPLICATION_PAGE_IR_SUB view can help identifying the subscription ID to remove.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SUBSCRIPTION (
        p_subscription_id => 12345);
END;
```

32.25 REMOVE_WEB_ENTRY_POINT Procedure

Removes a public procedure from the list of allowed objects that can be called via the URL.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_WEB_ENTRY_POINT (
    p_name    IN VARCHAR2 )
```

Parameters

Parameter	Description
p_name	The procedure name, prefixed by package name and schema, unless a public synonym exists.

Examples

Prevent myschema.mypkg.proc from being called via POST requests.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_WEB_ENTRY_POINT (
        p_name    'MYSCHHEMA.MYPKG.PROC' );
    COMMIT;
END;
```

32.26 REMOVE_WORKSPACE Procedure

This procedure removes a workspace from an Oracle APEX 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

Parameter	Description
p_workspace	The name of the workspace to be removed.
p_drop_users	Y to drop the database user associated with the workspace. The default is N.
p_drop_tablespaces	Y to drop the tablespace associated with the database user associated with the workspace. The default is N.

Example

The following example demonstrates how to use the REMOVE_WORKSPACE procedure to remove an existing workspace named MY_WORKSPACE, along with the associated database users and tablespace.

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

32.27 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

Parameter	Description
p_workspace	The workspace.

Example

This example removes all exceptions that allow the assignment of restricted schemas to HR_WORKSPACE.

```
BEGIN
    apex_instance_admin.remove_schema_exceptions (
        p_workspace => 'HR_WORKSPACE' );
    COMMIT;
END;
```

 See Also:

- [CREATE_SCHEMA_EXCEPTION Procedure](#)
- [RESTRICT_SCHEMA Procedure](#)
- [UNRESTRICT_SCHEMA Procedure](#)
- [REMOVE_SCHEMA_EXCEPTION Procedure](#)
- [REMOVE_SCHEMA_EXCEPTIONS Procedure](#)

32.28 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

Parameter	Description
p_workspace_id	The unique ID of the workspace.

Example

This example demonstrates setting up two separate Oracle APEX 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 APEX 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 APEX instance 2, ensure that application IDs are between 200000 and 299999.

```
begin
    apex_instance_admin.set_parameter('APPLICATION_ID_MIN', 200000);
    apex_instance_admin.set_parameter('APPLICATION_ID_MAX', 299999);
end;
```

3. Later, the operations team decides that workspace MY_WORKSPACE with ID 1234567890 should be moved from instance 1 to instance 2. The required steps are:
 - a. Export the workspace, applications and data on instance 1 (not shown here).
 - b. Ensure that no other application on instance 1 can reuse application IDs of this workspace.

```
begin
    apex_instance_admin.reserve_workspace_app_ids(1234567890);
end;
```

- c. Drop workspace, accompanying data and users on instance 1.

```
begin
    apex_instance_admin.remove_workspace('MY_WORKSPACE');
end;
```

- d. Import the workspace, applications and data on instance 2 (not shown here).

 **See Also:**

To undo a reservation, see [FREE_WORKSPACE_APP_IDS Procedure](#).

32.29 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

Parameter	Description
p_schema	The schema.

Example

This example revokes the privilege to assign schema HR to workspaces.

```
BEGIN
    apex_instance_admin.restrict_schema(p_schema => 'HR');
    COMMIT;
END;
```

See Also:

- [CREATE_SCHEMA_EXCEPTION Procedure](#)
- [UNRESTRICT_SCHEMA Procedure](#)
- [REMOVE_SCHEMA_EXCEPTION Procedure](#)
- [REMOVE_SCHEMA_EXCEPTIONS Procedure](#)
- [REMOVE_WORKSPACE_EXCEPTIONS Procedure](#)

32.30 REVOKE_EXTENSION_WORKSPACE Procedure

This procedure revokes an existing grant from an extension workspace.

Syntax

```
APEX_INSTANCE_ADMIN.REVOKE_EXTENSION_WORKSPACE (
    p_from_workspace      IN VARCHAR2,
    p_to_workspace        IN VARCHAR2 )
```

Parameters

Parameter	Description
p_from_workspace	Name of workspace granting access.
p_to_workspace	Name of extension workspace.

Example

The following example revokes grant from extension workspace EXTENSIONS to workspace MY_WORKSPACE.

```
BEGIN
    apex_instance_admin.revoke_extension_workspace(
        p_from_workspace      => 'MY_WORKSPACE',
        p_to_workspace        => 'EXTENSIONS' );
END;
```

32.31 SET_LOG_SWITCH_INTERVAL Procedure

Set the log switch interval for each of the logs maintained by Oracle APEX.

Syntax

```
APEX_INSTANCE_ADMIN.SET_LOG_SWITCH_INTERVAL(
    p_log_name           IN VARCHAR2,
    p_log_switch_after_days IN NUMBER );
```

Parameters

Parameters	Description
p_log_name	Specifies the name of the log. Valid values include ACCESS, ACTIVITY, AUTOMATION, CLICKTHRU, DEBUG, WEBSERVICE, and WEBSOURCESYNC.
p_log_switch_after_days	This interval must be a positive integer between 1 and 180.

Example

This example sets the log switch interval for the ACTIVITY log to 30 days.

```
BEGIN
    apex_instance_admin.set_log_switch_interval( p_log_name => 'ACTIVITY',
p_log_switch_after_days => 30 );
    COMMIT;
END;
```

32.32 SET_PARAMETER Procedure

This 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

Parameter	Description
p_parameter	The instance parameter to be set.
p_value	The value of the parameter. 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 APEX instance.

```
BEGIN
    APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_HOST_ADDRESS',
'mail.example.com');
    COMMIT;
END;
```

32.33 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

Parameters	Description
p_workspace	This is the name of the workspace for which the resource consumer group is to be set.
p_rm_consumer_group	The parameter P_RM_CONSUMER_GROUP is the Oracle Database Resource Manager Consumer Group name. The consumer group does not have to exist at the time this procedure is invoked. But if the Resource Manager Consumer Group is set for a workspace and the consumer group does not exist, then an error will be raised when anyone attempts to login to this workspace or execute any application in the workspace. If the value of P_RM_CONSUMER_GROUP is null, then the Resource Manager consumer group associated with the specified workspace is cleared.

Example

The following example sets the workspace to the Resource Manager consumer group "CUSTOM_GROUP1":

```
BEGIN
    apex_instance_admin.set_workspace_consumer_group(
        p_workspace => 'MY_WORKSPACE',
        p_rm_consumer_group => 'CUSTOM_GROUP1' );
    COMMIT;
END;
/
```

32.34 SET_WORKSPACE_PARAMETER Procedure

This procedure sets the designated workspace parameter.

Syntax

```
APEX_INSTANCE_ADMIN.SET_WORKSPACE_PARAMETER (
    p_workspace      IN VARCHAR2,
    p_parameter     IN VARCHAR2,
    p_value         IN VARCHAR2 )
```

Parameters

Parameter	Description
p_workspace	The name of the workspace to which you are setting the workspace parameter.
p_parameter	The parameter name which overrides the instance parameter value of the same for this workspace. Parameter names include:
	<ul style="list-style-type: none"> • ACCOUNT_LIFETIME_DAYS • ALLOW_HOSTING_EXTENSIONS • ALLOW_HOSTNAMES • CONTENT_CACHE_SIZE_TARGET • CONTENT_CACHE_MAX_FILE_SIZE • ENV_BANNER_COLOR • ENV_BANNER_LABEL • ENV_BANNER_POS • ENV_BANNER_ENABLE • EXPIRE_FND_USER_ACCOUNTS • MAX_LOGIN_FAILURES • MAX_SESSION_IDLE_SEC • MAX_SESSION_LENGTH_SEC • MAX_WEBSERVICE_REQUESTS • PATH_PREFIX • QOS_MAX_WORKSPACE_REQUESTS • QOS_MAX_SESSION_REQUESTS • QOS_MAX_SESSION_KILL_TIMEOUT • RM_CONSUMER_GROUP • SESSION_TIMEOUT_WARNING_SEC • WEBSERVICE_LOGGING • WORKSPACE_EMAIL_MAXIMUM • WORKSPACE_MAX_FILE_BYTES
p_value	The parameter value.

Example

The following example demonstrates how to use the `set_workspace_parameter` procedure to restrict URLs for accessing applications in the HR workspace that have `hr.example.com` in the hostname or domain name.

```

BEGIN
    apex_instance_admin.set_workspace_parameter (
        p_workspace => 'HR',
        p_parameter => 'ALLOW_HOSTNAMES',
        p_value      => 'hr.example.com' );
    COMMIT;
END;

```

 See Also:

- Available Parameter Values

32.35 TRUNCATE_LOG Procedure

Truncates the log entries specified by the input parameter.

Syntax

```
APEX_INSTANCE_ADMIN.TRUNCATE_LOG (
    p_log      IN VARCHAR2 )
```

Parameters

Parameter	Description
p_log	This parameter can have one of the following values: <ul style="list-style-type: none"> • ACTIVITY - removes all entries that record page access. • AUTOMATION - removes all entries from the automation logs. • CLICKS - removes all entries that record clicks tracked to external sites. • DEBUG - removes all entries captured during debug sessions. • FILE - removes all entries that record automatic file purge activity. • 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. • MAIL - removes all entries that record mail sent. • PURGE - removes all entries that record automatic workspace purge activity. • REST_SYNCHRONIZATION - removes all entries to record REST Source Synchronizations. • 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 • USER_ACCESS - removes all entries that record user login. • WEB_SERVICES - removes all entries that record web service calls initiated from this APEX instance. • WORKSPACE_HIST - removes all entries that record daily workspace summary.

Example

The following example demonstrates how to use the TRUNCATE_LOG procedure to remove all log entries that record access to APEX application pages.

```
BEGIN
    APEX_INSTANCE_ADMIN.TRUNCATE_LOG ('ACTIVITY');
END;
```

32.36 UNLOCK_USER Procedure

This procedure unlocks an Oracle APEX workspace user account and optionally also changes the user's password.

Syntax

```
APEX_INSTANCE_ADMIN.UNLOCK_USER (
    p_workspace    IN VARCHAR2,
    p_username     IN VARCHAR2,
    p_password     IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_workspace	Workspace of the user.
p_username	Name of the user.
p_password	New password. If not set, only unlocks the account.

Example 1

The following example unlocks the user `ADMIN` in the instance administration workspace and changes the password.

```
BEGIN
    apex_instance_admin.unlock_user (
        p_workspace => 'INTERNAL',
        p_username  => 'ADMIN',
        p_password   => 'example password' );
    COMMIT;
END;
```

Example 2

The following example unlocks the user `EXAMPLE_USER` in `SOME_WORKSPACE` **without** updating the password.

```
BEGIN
    apex_instance_admin.unlock_user (
        p_workspace => 'SOME_WORKSPACE',
        p_username  => 'EXAMPLE_USER' );
    COMMIT;
END;
```

32.37 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

Parameter	Description
p_schema	The schema.

Example

This example re-grants the privilege to assign schema HR to workspaces.

```
BEGIN
    apex_instance_admin.unrestrict_schema(p_schema => 'HR');
    COMMIT;
END;
```

 **See Also:**

- [CREATE_SCHEMA_EXCEPTION Procedure](#)
- [RESTRICT_SCHEMA Procedure](#)
- [REMOVE_SCHEMA_EXCEPTION Procedure](#)
- [REMOVE_SCHEMA_EXCEPTIONS Procedure](#),
- [REMOVE_WORKSPACE_EXCEPTIONS Procedure](#)

32.38 VALIDATE_EMAIL_CONFIG Procedure

This procedure attempts to establish a connection with the email server configured in an Oracle APEX instance. An error is raised if the connection is unsuccessful. This can indicate incorrect SMTP instance parameters, missing Network ACL, missing SSL certificate in Oracle Wallet, or a problem on the email server side. Correct the instance configuration and re-execute this procedure to confirm.

This procedure exits if the connection successfully establishes.

Syntax

```
APEX_INSTANCE_ADMIN.VALIDATE_EMAIL_CONFIG
```

Parameters

None.

Example

```
BEGIN
    APEX_INSTANCE_ADMIN.VALIDATE_EMAIL_CONFIG;
END;
```

 **See Also:**

- [APEX_MAIL](#)
- Configuring Email in *Oracle APEX Administration Guide*

APEX_IG

The APEX_IG package provides utilities you can use when programming in the Oracle APEX environment related to interactive grids. You can use the APEX_IG package to add filters, reset or clear report settings, delete saved reports and change report owners.

- [ADD_FILTER Procedure Signature 1](#)
- [ADD_FILTER Procedure Signature 2](#)
- [CHANGE_REPORT_OWNER Procedure](#)
- [CLEAR_REPORT Procedure Signature 1](#)
- [CLEAR_REPORT Procedure Signature 2](#)
- [DELETE_REPORT Procedure](#)
- [GET_LAST_VIEWED_REPORT_ID Function](#)
- [RESET_REPORT Procedure Signature 1](#)
- [RESET_REPORT Procedure Signature 2](#)

33.1 ADD_FILTER Procedure Signature 1

This procedure creates a filter on an interactive grid 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 grid reloads the page with download format in the REQUEST value. Any interactive grid settings changes (such as add filter or reset report) are done in an Ajax request. 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_IG.ADD_FILTER (
    p_page_id          IN NUMBER,
    p_region_id        IN NUMBER,
    p_filter_value     IN VARCHAR2,
    p_column_name      IN VARCHAR2 DEFAULT NULL,
    p_operator_abbr    IN VARCHAR2 DEFAULT NULL,
    p_is_case_sensitive IN BOOLEAN DEFAULT FALSE,
    p_report_id        IN NUMBER DEFAULT NULL )
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region (ID).
p_filter_value	The filter value. This value is not used for operator N and NN.
p_column_name	Name of the report SQL column, or column alias, to be filtered.
p_operator_abbr	Filter type. Valid values are as follows: EQ = Equals NEQ = Not Equals LT = Less than LTE = Less than or equal to GT = Greater Than GTE = Greater than or equal to N = Null NN = Not Null C = Contains NC = Not Contains IN = SQL In Operator NIN = SQL Not In Operator
p_is_case_sensitive	Case sensitivity of the row search filter. This value is not used for a column filter, where p_report_column is set. Valid values are as follows: <ul style="list-style-type: none">• TRUE• FALSE (This is the default value.)
p_report_id	The saved report ID within the current application page. If p_report_id is NULL, it adds the filter to the last viewed report settings.

Example 1

The following example shows how to use the ADD_FILTER procedure to filter the interactive grid with report ID of 901029800374639010 in page 1, region 3335704029884222 of the current application with DEPTNO equals 30

```
BEGIN
    APEX_IG.ADD_FILTER(
        p_page_id          => 1,
        p_region_id        => 3335704029884222,
        p_filter_value     => '30',
        p_column_name      => 'DEPTNO',
        p_operator_abbr   => 'EQ',
        p_report_id        => 901029800374639010);
END;
```

Example 2

The following example shows how to use the ADD_FILTER procedure to filter the interactive grid with report ID of 901029800374639010 in page 1, region 3335704029884222 of the current application with rows containing the case-sensitive word Salary.

```
BEGIN
    APEX_IG.ADD_FILTER(
        p_page_id      => 1,
        p_region_id    => 3335704029884222,
        p_filter_value => 'Salary',
        p_is_case_sensitive => true,
        p_report_id    => 901029800374639010);
END;
```

33.2 ADD_FILTER Procedure Signature 2

This procedure creates a filter on an interactive grid using a report name.

 **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 grid reloads the page with download format in the REQUEST value. Any interactive grid settings changes (such as add filter or reset report) are done in an Ajax request. 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_IG.ADD_FILTER (
    p_page_id      IN NUMBER,
    p_region_id    IN NUMBER,
    p_filter_value IN VARCHAR2,
    p_column_name  IN VARCHAR2 DEFAULT NULL,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_is_case_sensitive IN BOOLEAN DEFAULT FALSE,
    p_report_name  IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region (ID).
p_filter_value	This is the filter value. This value is not used for N and NN.
p_column_name	Name of the report SQL column, or column alias, to be filtered.

Parameter	Description
p_operator_abbr	Filter type. Valid values are as follows: EQ = Equals NEQ = Not Equals LT = Less than LTE = Less than or equal to GT = Greater Than GTE = Greater than or equal to N = Null NN = Not Null C = Contains NC = Not Contains IN = SQL In Operator NIN = SQL Not In Operator
p_is_case_sensitive	Case sensitivity of the row search filter. This value is not used for a column filter, where p_report_column is set. Valid values are as follows: <ul style="list-style-type: none">• TRUE• FALSE (This is the default value.)
p_report_name	The saved report name within the current application page. If p_report_name is NULL, it adds the filter to the last viewed report settings.

Example 1

The following example shows how to use the ADD_FILTER procedure to filter the interactive grid with report name of Statistics in page 1, region 3335704029884222 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_IG.ADD_FILTER(
        p_page_id          => 1,
        p_region_id        => 3335704029884222,
        p_filter_value     => '30',
        p_column_name      => 'DEPTNO',
        p_operator_abbr    => 'EQ',
        p_report_name       => 'Statistics');
END;
```

Example 2

The following example shows how to use the ADD_FILTER procedure to filter the interactive grid with report name of Statistics in page 1, region 3335704029884222 of the current application with rows containing the case-sensitive word Salary.

```
BEGIN
    APEX_IG.ADD_FILTER(
        p_page_id          => 1,
        p_region_id        => 3335704029884222,
        p_filter_value     => 'Salary',
```

```

    p_is_case_sensitive => true,
    p_report_name       => 'Statistics');
END;

```

33.3 CHANGE_REPORT_OWNER Procedure

This procedure changes the owner of a saved interactive grid report using a report ID. This procedure cannot change the owner of default interactive grid report.

Syntax

```
APEX_IG.CHANGE_REPORT_OWNER (
    p_application_id IN NUMBER DEFAULT apex_application.g_flow_id,
    p_report_id      IN NUMBER,
    p_old_owner      IN VARCHAR2,
    p_new_owner      IN VARCHAR2 )
```

Parameters

Parameters	Description
p_application_id	The application ID containing the interactive grid. If p_application_id is NULL, it defaults to the application ID in apex_application.g_flow_id.
p_report_id	The saved report ID within the current application page.
p_old_owner	The previous owner name to change from (case-sensitive). The owner needs to be a valid login user accessing the report.
p_new_owner	The new owner name to change to (case-sensitive). The owner must be a valid login user accessing the report.

Example

This example shows how to use CHANGE_REPORT_OWNER procedure to change the old owner name of JOHN to the new owner name of JOHN.DOE for a saved report. The saved report has a report ID of 1235704029884282 and resides in the application with ID 100.

```

BEGIN
    APEX_IG.CHANGE_REPORT_OWNER (
        P_application_id => 100,
        p_report_id      => 1235704029884282,
        p_old_owner      => 'JOHN',
        p_new_owner      => 'JOHN.DOE');
END;

```

33.4 CLEAR_REPORT Procedure Signature 1

This procedure clears report filter 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 grid reloads the page with download format in the REQUEST value. Any interactive grid settings changes (such as add filter or reset report) are done in an Ajax request. 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_IG.CLEAR_REPORT (
    p_page_id    IN NUMBER,
    p_region_id IN NUMBER,
    p_report_id  IN NUMBER DEFAULT NULL )
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region ID.
p_report_id	The saved report ID within the current application page. If p_report_id is NULL, it clears the last viewed report settings.

Example

The following example shows how to use the CLEAR_REPORT procedure signature 1 to reset interactive grid filter settings with report ID of 901029800374639010 in page 1, region 3335704029884222 of the current application.

```
BEGIN
    APEX_IG.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id   => 3335704029884222,
        p_report_id   => 901029800374639010);
END;
```

33.5 CLEAR_REPORT Procedure Signature 2

This procedure clears filter report settings to the developer defined default settings using the report name.

 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 grid reloads the page with download format in the REQUEST value. Any interactive grid settings changes (such as add filter or reset report) are done in an Ajax request. 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_IG.CLEAR_REPORT (
    p_page_id      IN NUMBER,
    p_region_id    IN NUMBER,
    p_report_name  IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region (ID).
p_report_name	The saved report name within the current application page. If p_report_name is NULL, it clears the last viewed report settings.

Example

The following example shows how to use the CLEAR_REPORT procedure signature 2 to reset interactive grid filter settings with report name of `Statistics` in page 1, region 3335704029884222 of the current application.

```
BEGIN
    APEX_IG.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id    => 3335704029884222,
        p_report_name  => 'Statistics');
END;
```

33.6 DELETE_REPORT Procedure

This procedure deletes a saved interactive grid report. It deletes a specific saved report in the current logged in workspace and application.

Syntax

```
APEX_IG.DELETE_REPORT (
    p_application_id IN NUMBER DEFAULT apex_application.g_flow_id,
    p_report_id      IN NUMBER )
```

Parameters

Parameter	Description
p_application_id	The application ID containing the interactive grid. If p_application_id is NULL, it defaults to the application ID in apex_application.g_flow_id.
p_report_id	Report ID to delete within the current Oracle APEX application.

Example

The following example shows how to use the `DELETE_REPORT` procedure to delete the saved interactive grid report with ID 901029800374639010 in application ID 100.

```
BEGIN
    APEX_IG.DELETE_REPORT (
        P_application_id => 100,
        p_report_id      => 901029800374639010);
END;
```

33.7 GET_LAST_VIEWED_REPORT_ID Function

This function returns the last viewed base report ID of the specified page and region.

Syntax

```
APEX_IG.GET_LAST_VIEWED_REPORT_ID (
    p_page_id    IN NUMBER,
    p_region_id IN NUMBER )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region ID.

Example

The following example shows how to use the `GET_LAST_VIEWED_REPORT_ID` function to retrieve the last viewed report ID in page 1, region 3335704029884222 of the current application.

```
DECLARE
    l_report_id number;
BEGIN
    l_report_id := APEX_IG.GET_LAST_VIEWED_REPORT_ID (
        p_page_id    => 1,
        p_region_id => 3335704029884222);
END;
```

33.8 RESET_REPORT Procedure Signature 1

This procedure resets report settings to the developer defined default settings using the report ID.

Syntax

```
APEX_IG.RESET_REPORT (
    p_page_id    IN NUMBER,
    p_region_id  IN NUMBER,
    p_report_id  IN NUMBER DEFAULT NULL );
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region ID.
p_report_name	The saved report name within the current application page. If p_report_id is NULL, it resets the last viewed report settings.

Example

The following example shows how to use the RESET_REPORT procedure signature 1 to reset interactive grid settings with report ID of 901029800374639010 in page 1, region 3335704029884222 of the current application.

```
BEGIN
    APEX_IG.RESET_REPORT(
        p_page_id      => 1,
        p_region_id   => 3335704029884222,
        p_report_id   => 901029800374639010);
END;
```

33.9 RESET_REPORT Procedure Signature 2

This procedure resets report settings to the developer defined default settings using the report name.

Syntax

```
APEX_IG.RESET_REPORT (
    p_page_id      IN NUMBER,
    p_region_id   IN NUMBER,
    p_report_name IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive grid.
p_region_id	The interactive grid region ID.
p_report_name	The saved report name within the current application page. If p_report_name is NULL, it resets the last viewed report settings.

Example

The following example shows how to use the RESET_REPORT procedure signature 2 to reset interactive grid settings with report name of Statistics in page 1, region 3335704029884222 of the current application.

```
BEGIN
    APEX_IG.RESET_REPORT(
        p_page_id      => 1,
        p_region_id   => 3335704029884222,
        p_report_name => 'Statistics' );
END;
```

APEX_IR

The APEX_IR package provides utilities you can use when programming in the Oracle APEX 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.

- Constants and Data Types
- ADD_FILTER Procedure Signature 1
- ADD_FILTER Procedure Signature 2
- CHANGE_REPORT_OWNER Procedure
- CHANGE_SUBSCRIPTION_EMAIL Procedure
- CHANGE_SUBSCRIPTION_LANG Procedure
- CLEAR_REPORT Procedure Signature 1
- CLEAR_REPORT Procedure Signature 2
- CLONE_REPORT Function
- DELETE_REPORT Procedure
- DELETE_SUBSCRIPTION Procedure
- EXPORT_SAVED_REPORTS Function
- GET_LAST_VIEWED_REPORT_ID Function
- GET_REPORT Function (Deprecated)
- IMPORT_SAVED_REPORTS Procedure
- RESET_REPORT Procedure Signature 1
- RESET_REPORT Procedure Signature 2

34.1 Constants and Data Types

Constants

The APEX_IR package uses the following constants.

```
c_view_report      constant VARCHAR2(20) := 'REPORT';
c_view_groupby    constant varchar2(20) := 'GROUP_BY';
c_view_pivot      constant varchar2(20) := 'PIVOT';
```

Data Types

The APEX_IR package uses the following data types.

```
type t_report is record(
    sql_query  varchar2(32767),
```

```
        binds      apex_plugin_util.t_bind_list)
);
```

34.2 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region (ID).
p_report_column	Name of the report SQL column, or column alias, to be filtered.
p_filter_value	The filter value. This value is not used for N and NN.
	Enter multiple values in a comma-separated list. Enclose multiple filter values separated by commas in backslash characters (\). For example, if the p_operator_abbr is type IN or NIN, and you wish to filter for the values CLOSED and OPEN, then set p_filter_value to \CLOSED,OPEN\.

Parameter	Description
p_operator_abbr	Filter type. Valid values are as follows: EQ = Equals NEQ = Not Equals LT = Less than LTE = Less then or equal to GT = Greater Than GTE = Greater than or equal to LIKE = SQL Like operator NLIKE = Not Like N = Null NN = Not Null C = Contains NC = Not Contains IN = SQL In Operator NIN = SQL Not In Operator
p_report_id	The saved report ID within the current application page. If p_report_id is NULL, it adds the filter to the last viewed report settings.

Example

The following example shows how to use the ADD_FILTER procedure to filter the interactive report with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_IR.ADD_FILTER(
        p_page_id      => 1,
        p_region_id   => 2505704029884282,
        p_report_column => 'DEPTNO',
        p_filter_value => '30',
        p_operator_abbr => 'EQ',
        p_report_id     => 880629800374638220);
END;
```

34.3 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region (ID).
p_report_column	Name of the report SQL column, or column alias, to be filtered.
p_filter_value	This is the filter value. This value is not used for N and NN.
p_operator_abbr	Filter type. Valid values are as follows: EQ = Equals NEQ = Not Equals LT = Less than LTE = Less then or equal to GT = Greater Than GTE = Greater than or equal to LIKE = SQL Like operator NLIKE = Not Like N = Null NN = Not Null C = Contains NC = Not Contains IN = SQL In Operator NIN = SQL Not In Operator
p_report_alias	The saved report alias within the current application page. If p_report_alias is NULL, it adds filter to the last viewed report settings.

Example

The following example shows how to use the ADD_FILTER procedure to filter an interactive report with a report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_IR.ADD_FILTER(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_column => 'DEPTNO',
        p_filter_value  => '30',
```

```

    p_operator_abbr => 'EQ',
    p_report_alias  => 'CATEGORY_REPORT');
END;

```

34.4 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

Parameters	Description
p_report_id	The saved report ID within the current application page.
p_old_owner	The previous owner name to change from (case sensitive). The owner needs to a valid login user accessing the report.
p_new_owner	The new owner name to change to (case sensitive). The owner must be a valid login user accessing the report.

Example

This example shows how to use `CHANGE_REPORT_OWNER` procedure to change the old owner name of *JOHN* to the new owner name of *JOHN.DOE* for a saved report. The saved report has a report ID of 1235704029884282.

```
BEGIN
    APEX_IR.CHANGE_REPORT_OWNER (
        p_report_id      => 1235704029884282,
        p_old_owner     => 'JOHN',
        p_new_owner     => 'JOHN.DOE');
END;
```

34.5 CHANGE_SUBSCRIPTION_EMAIL Procedure

This procedure changes the interactive report subscription's email address. When an email is sent out, the subscription sends a message to the defined email address.

Syntax

```
APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (
    p_subscription_id   IN NUMBER,
    p_email_address     IN VARCHAR2 );
```

Parameters

Parameter	Description
p_subscription_id	Subscription ID to change the email address within the current workspace.
p_email_address	The new email address to change to. The email address needs to be a valid email syntax and cannot be set to null.

Example

The following example shows how to use the CHANGE_SUBSCRIPTION_EMAIL procedure to change the email address to `some.user@example.com` for the interactive report subscription 956136850459718525.

```
BEGIN
    APEX_IR.CHANGE_SUBSCRIPTION_EMAIL (
        p_subscription_id => 956136850459718525,
        p_email_address   => 'some.user@example.com');
END;
```

34.6 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

Parameter	Description
p_subscription_id	Subscription ID to change the language within the current workspace.
p_language	This is an IANA language code. Some examples include: en, de, de-at, zh-cn, and pt-br.

Example

The following example shows how to use the CHANGE_SUBSCRIPTION_LANG procedure to change the subscription with the ID of 567890123 to German in the current workspace.

```
BEGIN
    APEX_IR.CHANGE_SUBSCRIPTION_LANG(
        p_subscription_id => 567890123,
        p_language         => 'de');
END;
```

34.7 CLEAR_REPORT Procedure Signature 1

This procedure clears report settings using the report ID.

 **Note:**

The use of this procedure in a page rendering process causes report download issues (CSV, HTML, Email, and so on). When a user downloads the report, the interactive report reloads the page with download format in the REQUEST value. Any interactive report settings changes (such as add filter or reset report) are done in partial page refresh. Thus, the download data may not match the report data user is seeing. For this reason, Oracle recommends only using this procedure in a page submit process.

Syntax

```
APEX_IR.CLEAR_REPORT (
    p_page_id    IN NUMBER,
    p_region_id  IN NUMBER,
    p_report_id  IN NUMBER DEFAULT NULL )
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region (ID).
p_report_id	The saved report ID within the current application page. If p_report_id is NULL, it clears the last viewed report settings.

Example

The following example shows how to use the CLEAR_REPORT procedure to clear interactive report settings with a report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id   => 2505704029884282,
        p_report_id   => 880629800374638220);
END;
```

34.8 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region (ID).
p_report_alias	The saved report alias within the current application page. If p_report_alias is NULL, it clears the last viewed report settings.

Example

The following example shows how to use the CLEAR_REPORT procedure to clear interactive report settings with report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.CLEAR_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_alias => 'CATEGORY_REPORT');
END;
```

34.9 CLONE_REPORT Function

This function clones a user-saved report and returns a new report ID.

You can clone into a Private or Public report, but you **cannot** clone into a default report.

Syntax

```
APEX_IR.CLONE_REPORT (
    p_report_id      IN NUMBER,
    p_new_name       IN VARCHAR2,
    p_new_description IN VARCHAR2 DEFAULT NULL,
```

```

    p_new_owner      IN VARCHAR2 DEFAULT apex_application.g_user,
    p_new_is_public IN BOOLEAN  DEFAULT FALSE,
    p_replace_report IN BOOLEAN  DEFAULT TRUE )
RETURN NUMBER;

```

Parameters

Parameter	Description
p_report_id	The source report ID to clone.
p_new_name	The new report name.
p_new_description	The new report description.
p_new_owner	The case-sensitive new owner of the report. If not passed, current user is the owner.
p_new_is_public	If new report is Public. If not passed, clones as Private report.
p_replace_report	If TRUE (default), report will be replaced if exists. If FALSE, an error raises if a report with the same name and owner already exists.

Example

The following example clones a report ID selected from a page item value. The report name and owner are overwritten by the parameter values, and the report is cloned as public report.

```

DECLARE
    l_new_report_id number;
BEGIN
    l_new_report_id := apex_ir.clone_report (
        p_report_id      => :P1_REPORT_ID,
        p_new_name       => 'New Cloned Report',
        p_new_owner      => :APP_USER,
        p_new_is_public  => true );
END;

```

34.10 DELETE_REPORT Procedure

This procedure deletes saved interactive reports. The deleted saved report is removed from the current logged-in workspace and application.

Syntax

```
APEX_IR.DELETE_REPORT (
    p_report_id IN NUMBER )
```

Parameters

Parameter	Description
p_report_id	Report ID to delete within the current Oracle APEX application.

Example

The following example shows how to use the `DELETE_REPORT` procedure to delete the saved interactive report with ID of 880629800374638220 in the current application.

```
BEGIN
    APEX_IR.DELETE_REPORT (
        p_report_id => 880629800374638220);
END;
```

34.11 DELETE_SUBSCRIPTION Procedure

This procedure deletes interactive report subscriptions.

Syntax

```
APEX_IR.DELETE_SUBSCRIPTION (
    p_subscription_id IN NUMBER )
```

Parameters

Parameter	Description
<code>p_subscription_id</code>	Subscription ID to delete within the current workspace.

Example

The following example shows how to use the `DELETE_SUBSCRIPTION` procedure to delete the subscription with ID of 567890123 in the current workspace.

```
BEGIN
    APEX_IR.DELETE_SUBSCRIPTION(
        p_subscription_id => 567890123);
END;
```

34.12 EXPORT_SAVED_REPORTS Function

This function exports multiple saved reports from the current app and workspace. Exports default or user-saved reports.

If calling outside of Oracle APEX, use `apex_util.set_workspace` to set the current workspace.

Syntax

```
APEX_IR.EXPORT_SAVED_REPORTS (
    p_report_ids          IN apex_t_number,
    p_credential_static_id IN VARCHAR2 )
RETURN CLOB;
```

Parameters

Parameter	Description
p_report_ids	The array of report IDs to export.
p_credential_static_id	The Key Pair authentication credential static ID. This credential is used to create a signature for the export.
	Create compatible public and private keys using OpenSSH, and use those to create a Key Pair workspace web credential.

Returns

The signed and base64-encoded report export JSON object in CLOB.

Example

The following example exports report IDs (111111, 222222) from the current workspace using my_API_key_pair credential.

```
DECLARE
    l_export_clob clob;
BEGIN
    l_export_clob := apex_ir.export_saved_reports (
                    p_report_ids          => apex_t_number(
                                         111111, 222222 ),
                    p_credential_static_id => 'my_API_key_pair' );
END;
```

 **See Also:**

[SET_WORKSPACE Procedure](#)

34.13 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 )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.

Parameter	Description
p_region_id	The interactive report region ID.

Example

The following example shows how to use the GET_LAST_VIEWED_REPORT_ID function to retrieve the last viewed report ID in page 1, region 2505704029884282 of the current application.

```
DECLARE
    l_report_id number;
BEGIN
    l_report_id := APEX_IR.GET_LAST_VIEWED_REPORT_ID (
        p_page_id    => 1,
        p_region_id  => 2505704029884282);
END;
```

34.14 GET_REPORT Function (Deprecated)

 **Note:**

This function is deprecated and will be removed in a future release.

Use [OPEN_QUERY_CONTEXT Function](#) in APEX_REGION instead.

This function returns an interactive report runtime query.

Syntax

```
APEX_IR.GET_REPORT(
    p_page_id    IN NUMBER,
    p_region_id  IN NUMBER,
    p_report_id  IN NUMBER  DEFAULT NULL,
    p_view       IN VARCHAR2 DEFAULT c_view_report )
RETURN t_report;
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region ID.
p_report_id	The saved report ID within the current application page. If p_report_id is NULL, retrieves last viewed report query.
p_view	The view type available for the report. The values can be APEX_IR.C_VIEW_REPORT, APEX_IR.C_VIEW_GROUPBY, or APEX_IR.C_VIEW_PIVOT. If p_view is NULL, retrieves the view currently used by the report. If the p_view passed does not exist for the current report, an error raises.

Example 1

The following example shows how to use the GET_REPORT function to retrieve the runtime report query with bind variable information with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
DECLARE
    l_report apex_ir.t_report;
    l_query varchar2(32767);
BEGIN
    l_report := APEX_IR.GET_REPORT (
        p_page_id => 1,
        p_region_id => 2505704029884282,
        p_report_id => 880629800374638220);
    l_query := l_report.sql_query;
    sys.htp.p('Statement = '''||l_report.sql_query);
    for i in 1..l_report.binds.count
    loop
        sys.htp.p(i||'. ''||l_report.binds(i).name||' = '||l_report.binds(i).value);
    end loop;
END;
```

Example 2

The following example shows how to use the GET_REPORT function to retrieve Group By view query defined in the current report page with region 2505704029884282.

```
DECLARE
    l_report APEX_IR.T_REPORT;
BEGIN
    l_report := APEX_IR.GET_REPORT (
        p_page_id      => :APP_PAGE_ID,
        p_region_id   => 2505704029884282,
        p_view         => APEX_IR.C_VIEW_GROUPBY );
    sys.htp.p( 'Statement = '''||l_report.sql_query );
END;
```

 **See Also:**

[OPEN_QUERY_CONTEXT Function](#)

34.15 IMPORT_SAVED_REPORTS Procedure

This procedure imports saved reports into an app in the current workspace. Supports importing default or user-saved reports.

If calling outside of Oracle APEX, use `apex_util.set_workspace` to set the current workspace.

Syntax

```
APEX_IR.IMPORT_SAVED_REPORTS (
    p_export_content      IN CLOB,
    p_credential_static_id IN VARCHAR2,
    p_replace_report      IN BOOLEAN DEFAULT TRUE,
    p_new_owner            IN VARCHAR2 DEFAULT apex_application.g_user,
    p_new_application_id   IN NUMBER DEFAULT NULL );
```

Parameters

Parameter	Description
p_export_content	The signed and base64-encoded report export JSON.
p_credential_static_id	The Key Pair authentication credential static ID. The same credential used to sign the export content is used to verify.
p_replace_report	If TRUE (default), report is replaced if exists.
p_new_owner	The case-sensitive new owner of the reports. Only non-default reports can be overwritten with p_new_owner.
p_new_application_id	The new application ID of the reports. The reports are imported to the application containing valid interactive report regions.

Example

The following example imports reports using the uploaded export file and my_API_key_pair credential. The owner and application ID of the reports are overwritten by the entered page item values during the import.

```
DECLARE
    l_blob blob;
BEGIN
    SELECT blob_content
        INTO l_blob
        FROM apex_application_temp_files
       WHERE name = :P1_FILE;

    apex_ir.import_saved_reports (
        p_export_content      => apex_util.blob_to_clob( l_blob ),
        p_credential_static_id => 'my_API_key_pair',
        p_new_owner            => :P1_NEW_OWNER,
        p_new_application_id   => :P1_NEW_APP_ID );
END;
```



See Also:

[SET_WORKSPACE Procedure](#)

34.16 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region ID.
p_report_id	The saved report ID within the current application page. If p_report_id is NULL, it resets the last viewed report settings.

Example

The following example shows how to use the RESET_REPORT procedure signature 1 to reset interactive report settings with report ID of 880629800374638220 in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.RESET_REPORT(
        p_page_id      => 1,
        p_region_id   => 2505704029884282,
        p_report_id   => 880629800374638220);
END;
```

34.17 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_region_id	The interactive report region ID.
p_report_alias	The saved report alias within the current application page. If p_report_alias is NULL, it resets the last viewed report settings.

Example

The following example shows how to use the RESET_REPORT procedure to reset interactive report settings with a report alias of CATEGORY_REPORT in page 1, region 2505704029884282 of the current application.

```
BEGIN
    APEX_IR.RESET_REPORT(
        p_page_id      => 1,
        p_region_id    => 2505704029884282,
        p_report_alias => 'CATEGORY_REPORT');
END;
```

APEX_ITEM (Legacy)

This API is designated as legacy.

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](#)

35.1 CHECKBOX2 Function

This function creates check boxes.

Syntax

```
APEX_ITEM.CHECKBOX2 (
    p_idx          IN      NUMBER,
    p_value        IN      VARCHAR2 DEFAULT NULL,
    p_attributes   IN      VARCHAR2 DEFAULT NULL,
```

```

    p_checked_values      IN      VARCHAR2 DEFAULT NULL,
    p_checked_values_delimiter IN      VARCHAR2 DEFAULT ':' ,
    p_item_id             IN      VARCHAR2 DEFAULT NULL,
    p_item_label           IN      VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;

```

Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example 1 creates F01 and 2 creates F02
p_value	Value of a check box, hidden field, or input form item
p_attributes	Controls the size of the text field
p_checked_values	Values to be checked by default
p_checked_values_delimiter	Delimits the values in the previous parameter, p_checked_values
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Examples of Default Check Box Behavior

The following example demonstrates how to create a selected check box for each employee in the emp table.

```

SELECT APEX_ITEM.CHECKBOX2(1,empno,'CHECKED') "Select",
       ename, job
  FROM   emp
 ORDER BY 1

```

The following example demonstrates how to have all check boxes for employees display without being selected.

```

SELECT APEX_ITEM.CHECKBOX2(1,empno) "Select",
       ename, job
  FROM   emp
 ORDER BY 1

```

The following example demonstrates how to select the check boxes for employees who work in department 10.

```

SELECT APEX_ITEM.CHECKBOX2(1,empno,DECODE(deptno,10,'CHECKED',NULL)) "Select",
       ename, job
  FROM   emp
 ORDER BY 1

```

The next example demonstrates how to select the check boxes for employees who work in department 10 or department 20.

```

SELECT APEX_ITEM.CHECKBOX2(1,deptno,NULL,'10:20',':') "Select",
       ename, job

```

```
FROM emp
ORDER BY 1
```

Creating an On-Submit Process

If you are using check boxes in your application, you might need to create an On Submit process to perform a specific type of action on the selected rows. For example, you could have a Delete button that uses the following logic:

```
SELECT APEX_ITEM.CHECKBOX2(1,empno) "Select",
       ename, job
  FROM emp
 ORDER by 1
```

Consider the following sample on-submit process:

```
FOR I in 1..APEX_APPLICATION.G_F01.COUNT LOOP
    DELETE FROM emp WHERE empno = to_number(APEX_APPLICATION.G_F01(i));
END LOOP;
```

The following example demonstrates how to create unselected checkboxes for each employee in the emp table, with a unique ID. This is useful for referencing records from within JavaScript code:

```
SELECT APEX_ITEM.CHECKBOX2(1,empno,NULL,NULL,NULL,'f01_#ROWNUM#') "Select",
       ename, job
  FROM emp
 ORDER BY 1
```

35.2 DATE_POPUP Function

Use this function with forms that include date fields. The DATE_POPUP function dynamically generates a date field that has a popup calendar button.

Syntax

```
APEX_ITEM.DATE_POPUP (
    p_idx          IN NUMBER,
    p_row          IN NUMBER,
    p_value        IN VARCHAR2 DEFAULT NULL,
    p_date_format IN DATE DEFAULT 'DD-MON-YYYY',
    p_size         IN NUMBER DEFAULT 20,
    p_maxlength   IN NUMBER DEFAULT 2000,
    p_attributes   IN VARCHAR2 DEFAULT NULL,
    p_item_id     IN VARCHAR2 DEFAULT NULL,
    p_item_label   IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02.
p_row	This parameter is deprecated. Anything specified for this value is ignored.
p_value	Value of a field item.
p_date_format	Valid database date format.
p_size	Controls HTML tag attributes (such as disabled).
p_maxlength	Determines the maximum number of enterable characters. Becomes the maxlength attribute of the <input> HTML tag.
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to use APEX_ITEM.DATE_POPUP to create popup calendar buttons for the `hiredate` column.

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno) ||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4, rownum, hiredate, 'dd-mon-yyyy') hd,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

See Also:

[Oracle Database SQL Language Reference](#) for more information about the `TO_CHAR` or `TO_DATE` functions

35.3 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

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02.
p_value	Value of a field item.
p_date_format	Valid database date format.
p_size	Controls HTML tag attributes (such as disabled).
p_maxlength	Determines the maximum number of enterable characters. Becomes the maxlength attribute of the <input> HTML tag.
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.
p_default_value	The default date which should be selected in datepicker calendar popup.
p_max_value	The Maximum date that can be selected from the datepicker.
p_min_value	The Minimum date that can be selected from the datepicker.
p_show_on	Determines when the datepicker displays, on button click or on focus of the item or both.
p_number_of_months	Determines number of months displayed. Value should be in this array format: [row, column]
p_navigation_list_for	Determines if a select list is displayed for Changing Month, Year, or Both. Possible values include: MONTH, YEAR, MONTH_AND_YEAR. Default NULL.
p_year_range	The range of years displayed in the year selection list.
p_validation_date	Used to store the Date value for the which date validation failed.

 See Also:

Oracle Database SQL Language Reference for information about the TO_CHAR or TO_DATE functions

35.4 DISPLAY_AND_SAVE Function

Use this function to display an item as text, but save its value to session state.

Syntax

```
APEX_ITEM.DISPLAY_AND_SAVE (
    p_idx      IN      NUMBER,
    p_value    IN      VARCHAR2 DEFAULT NULL,
    p_item_id  IN      VARCHAR2 DEFAULT NULL,
    p_item_label IN      VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02.
p_value	Current value.
p_item_id	HTML attribute ID for the tag.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to use the APEX_ITEM.DISPLAY_AND_SAVE function.

```
SELECT APEX_ITEM.DISPLAY_AND_SAVE(10,empno) c FROM emp
```

35.5 HIDDEN Function

This function dynamically generates hidden form items.

Syntax

```
APEX_ITEM.HIDDEN (
    p_idx      IN      NUMBER,
    p_value    IN      VARCHAR2 DEFAULT
    p_attributes IN      VARCHAR2 DEFAULT NULL,
    p_item_id  IN      VARCHAR2 DEFAULT NULL,
    p_item_label IN      VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Number to identify the item you want to generate. The number determines which G_FXX global is populated See also APEX_APPLICATION .
p_value	Value of the hidden input form item.
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.

Example

Typically, the primary key of a table is stored as a hidden column and used for subsequent update processing, for example:

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno) ||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

The previous query could use the following page process to process the results:

```
BEGIN
    FOR i IN 1..APEX_APPLICATION.G_F01.COUNT LOOP
        UPDATE emp
        SET
            ename=APEX_APPLICATION.G_F02(i),
            job=APEX_APPLICATION.G_F03(i),
            hiredate=to_date(APEX_APPLICATION.G_F04(i),'dd-mon-yyyy'),
            sal=APEX_APPLICATION.G_F05(i),
            comm=APEX_APPLICATION.G_F06(i)
        WHERE empno=to_number(APEX_APPLICATION.G_F01(i));
    END LOOP;
END;
```

Note that the G_F01 column (which corresponds to the hidden EMPNO) is used as the key to update each row.

35.6 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 fields 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`:

```
DBMS_CRYPTO.HASH (
    SRC => UTL_RAW.CAST_TO_RAW('my_string'),
    TYP => DBMS_CRYPTO.HASH_MD5 );
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network.

Syntax

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

Parameters

Parameter	Description
<code>p_value01</code>	Fifty available inputs. If no parameters are supplied, defaults to <code>NULL</code> .
<code>...</code>	
<code>p_value50</code>	
<code>p_col_sep</code>	String used to separate <code>p_value</code> inputs. Defaults to <code> </code> (pipe symbol).
<code>p_item_id</code>	ID of the HTML form item.

Example

This function generates hidden form elements with the name `fcs`. The values can subsequently be accessed by using the `APEX_APPLICATION.G_FCS` array.

```
SELECT APEX_ITEM.MD5_CHECKSUM(ename,job,sal) md5_cks,
       ename, job, sal
  FROM emp
```

35.7 MD5_HIDDEN Function

Use this function for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces a hidden form field with a MD5 checksum as value which is based on up to 50 inputs. `APEX_ITEM.MD5_HIDDEN` also produces an MD5 checksum using Oracle database `DBMS_CRYPTO`:

```
UTL_RAW.CAST_TO_RAW(DBMS_CRYPTO.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network.

Syntax

```
APEX_ITEM.MD5_HIDDEN (
    p_idx      IN      NUMBER,
    p_value01  IN      VARCHAR2 DEFAULT NULL,
    p_value02  IN      VARCHAR2 DEFAULT NULL,
    p_value03  IN      VARCHAR2 DEFAULT NULL,
    ...
    p_value50  IN      VARCHAR2 DEFAULT NULL,
    p_col_sep  IN      VARCHAR2 DEFAULT '|',
    p_item_id  IN      VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
<code>p_idx</code>	Indicates the form element to be generated. For example, 1 equals F01 and 2 equals F02. Typically the <code>p_idx</code> parameter is constant for a given column.
<code>p_value01...50</code>	Fifty available inputs. Parameters not supplied default to NULL.
<code>p_col_sep</code>	String used to separate <code>p_value</code> inputs. Defaults to the pipe symbol ().
<code>p_item_id</code>	ID of the HTML form item.

Example

The `p_idx` parameter specifies the FXX form element to be generated. In the following example, 7 generates F07. Also note that an HTML hidden form element is generated.

```
SELECT APEX_ITEM.MD5_HIDDEN(7,ename,job,sal)md5_h, ename, job, sal
FROM emp
```

35.8 POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application shared list of values (LOV). Like other available functions in the `APEX_ITEM` package, `POPUP_FROM_LOV` function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUP_FROM_LOV (
    p_idx          IN      NUMBER,
    p_value        IN      VARCHAR2 DEFAULT NULL,
    p_lov_name    IN      VARCHAR2,
```

```

    p_width          IN  VARCHAR2 DEFAULT NULL,
    p_max_length    IN  VARCHAR2 DEFAULT NULL,
    p_form_index    IN  VARCHAR2 DEFAULT '0',
    p_escape_html   IN  VARCHAR2 DEFAULT NULL,
    p_max_elements  IN  VARCHAR2 DEFAULT NULL,
    p_attributes    IN  VARCHAR2 DEFAULT NULL,
    p_ok_to_query   IN  VARCHAR2 DEFAULT 'YES',
    p_item_id       IN  VARCHAR2 DEFAULT NULL,
    p_item_label    IN  VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;

```

Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_name parameter.
p_lov_name	Named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 (rarely used).
p_escape_html	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 APEX 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. Replacements for special characters that require an escaped equivalent: <ul style="list-style-type: none"> • &lt; for < • &gt; for > • &amp; for & Range of values is YES and NO. If YES, special characters are escaped. This parameter is useful if you know your query returns invalid HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup from an LOV named DEPT_LOV.

```

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

```

35.9 POPUP_FROM_QUERY Function

Generates an HTML popup select list from a query. This 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

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_query parameter.
p_lov_query	SQL query that is expected to select two columns (a display column and a return column). For example:
	SELECT dname, deptno FROM dept
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used.
	Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle APEX 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.

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. <ul style="list-style-type: none"> • &lt; for < • &gt; for > • &amp; for & Range of values is YES and NO. If YES, special characters are escaped. This parameter is useful if you know your query returns invalid HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup select list from the emp table.

```
SELECT APEX_ITEM.POPUP_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept')
dt
FROM emp
```

35.10 POPUPKEY_FROM_LOV Function

This function generates a popup key select list from a shared list of values (LOV). Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_LOV function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUPKEY_FROM_LOV (
    p_idx          IN NUMBER,
    p_value        IN VARCHAR2 DEFAULT NULL,
    p_lov_name    IN VARCHAR2,
    p_width        IN VARCHAR2 DEFAULT NULL,
    p_max_length   IN VARCHAR2 DEFAULT NULL,
    p_form_index   IN VARCHAR2 DEFAULT '0',
    p_escape_html  IN VARCHAR2 DEFAULT NULL,
    p_max_elements IN VARCHAR2 DEFAULT NULL,
    p_attributes   IN VARCHAR2 DEFAULT NULL,
    p_ok_to_query  IN VARCHAR2 DEFAULT 'YES',
    p_item_id      IN VARCHAR2 DEFAULT NULL,
    p_item_label   IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Although the text field associated with the popup displays in the first column in the LOV query, the actual value is specified in the second column in the query.

Parameters

Parameter	Description
p_idx	<p>Identifies a form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column</p> <p>Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example:</p> <pre>SELECT APEX_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt, APEX_ITEM.HIDDEN(3,empno) eno FROM emp;</pre>
p_value	Indicates the current value. This value should be one of the values in the P_LOV_NAME parameter.
p_lov_name	Identifies a named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used.
	Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by APEX must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.
p_escape_html	<p>Replacements for special characters that require an escaped equivalent.</p> <ul style="list-style-type: none"> • &lt; for < • &gt; for > • &amp; for & <p>This parameter is useful if you know your query returns invalid HTML.</p>
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to generate a popup key select list from a shared list of values (LOV).

```
SELECT APEX_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt
  FROM emp
```

35.11 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

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column. Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example:
	<pre>SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt, APEX_ITEM.HIDDEN(3,empno) eno</pre>
p_value	Form element current value. This value should be one of the values in the P_LOV_QUERY parameter.
p_lov_query	LOV query used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used.
	Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different website). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle APEX 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.

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. <ul style="list-style-type: none"> • &lt; for < • &gt; for > • &amp; for & This parameter is useful if you know your query returns illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to generate a popup select list from a SQL query.

```
SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM
dept') dt
      apex_item.hidden(3,empno) eno
      FROM emp;
```

35.12 RADIOGROUP Function

This function generates a radio group from a SQL query.

Syntax

```
APEX_ITEM.RADIOGROUP (
    p_idx          IN  NUMBER,
    p_value        IN  VARCHAR2 DEFAULT NULL,
    p_selected_value IN  VARCHAR2 DEFAULT NULL,
    p_display      IN  VARCHAR2 DEFAULT NULL,
    p_attributes   IN  VARCHAR2 DEFAULT NULL,
    p_onblur       IN  VARCHAR2 DEFAULT NULL,
    p_onchange     IN  VARCHAR2 DEFAULT NULL,
    p_onfocus      IN  VARCHAR2 DEFAULT NULL,
    p_item_id      IN  VARCHAR2 DEFAULT NULL,
    p_item_label   IN  VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable is used. Valid range of values is 1 to 50. For example 1 creates F01 and 2 creates F02.
p_value	Value of the radio group.

Parameter	Description
p_selected_value	Value that should be selected.
p_display	Text to display next to the radio option.
p_attributes	Extra HTML parameters you want to add.
p_onblur	JavaScript to execute in the onBlur event.
p_onchange	JavaScript to execute in the onChange event.
p_onfocus	JavaScript to execute in the onFocus event.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to select department 20 from the dept table as a default in a radio group.

```
SELECT APEX_ITEM.RADIOGROUP (1,deptno,'20',dname) dt
FROM   dept
ORDER  BY 1
```

35.13 SELECT_LIST Function

This function dynamically generates a static select list. This function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST (
    p_idx          IN  NUMBER,
    p_value        IN  VARCHAR2 DEFAULT NULL,
    p_list_values  IN  VARCHAR2 DEFAULT NULL,
    p_attributes   IN  VARCHAR2 DEFAULT NULL,
    p_show_null    IN  VARCHAR2 DEFAULT 'NO',
    p_null_value   IN  VARCHAR2 DEFAULT '%NULL%',
    p_null_text    IN  VARCHAR2 DEFAULT '%',
    p_item_id      IN  VARCHAR2 DEFAULT NULL,
    p_item_label   IN  VARCHAR2 DEFAULT NULL,
    p_show_extra   IN  VARCHAR2 DEFAULT 'YES' )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.
p_value	Current value. This value should be a value in the P_LIST_VALUES parameter.
p_list_values	List of static values separated by commas. Displays values and returns values that are separated by semicolons. Note that this is only available in the SELECT_LIST function.
p_attributes	Extra HTML parameters you want to add.

Parameter	Description
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example 1

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
```

Example 2

The following example generates 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;

```

35.14 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_XL, but its return value is VARCHAR2. Returned values are limited to 32k.

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

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_lov parameter.
p_lov	Text name of an application list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV(2,job,'JOB_FLOW_LOV') job
FROM emp
```

35.15 SELECT_LIST_FROM_LOV_XL Function

This function dynamically generates very large select lists (greater than 32K) from a shared list of values (LOV). Use this function in SQL queries where you need to handle a column value longer than 4000 characters.

Similar to other functions available in the `APEX_ITEM` package, these select list functions are designed to generate forms with F01 to F50 form array elements.

This function is the same as `SELECT_LIST_FROM_LOV`, but its return value is CLOB.

Syntax

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

Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_lov parameter.
p_lov	Text name of a list of values. This list of values must be defined in your application. This parameter is used only by the <code>select_list_from_lov</code> function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.

Parameter	Description
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates how to create a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV_XL(2,job,'JOB_FLOW_LOV') job
FROM emp
```

35.16 SELECT_LIST_FROM_QUERY Function

This function dynamically generates a select list from a query. Similar to other functions available in the `APEX_ITEM` package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_QUERY (
    p_idx          IN  NUMBER,
    p_value        IN  VARCHAR2 DEFAULT NULL,
    p_query        IN  VARCHAR2,
    p_attributes   IN  VARCHAR2 DEFAULT NULL,
    p_show_null    IN  VARCHAR2 DEFAULT 'YES',
    p_null_value   IN  VARCHAR2 DEFAULT '%NULL%',
    p_null_text    IN  VARCHAR2 DEFAULT '%',
    p_item_id      IN  VARCHAR2 DEFAULT NULL,
    p_item_label   IN  VARCHAR2 DEFAULT NULL,
    p_show_extra   IN  VARCHAR2 DEFAULT 'YES' )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_query parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example:
	SELECT dname, deptno FROM dept
	Note that this is used only by the <code>SELECT_LIST_FROM_QUERY</code> function.
	Also note, if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.

Parameter	Description
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY(3,job,'SELECT DISTINCT job FROM
emp')job
FROM emp
```

35.17 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

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_query parameter.

Parameter	Description
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: SELECT dname, deptno FROM dept
	Note that this is used only by the SELECT_LIST_FROM_QUERY_XL function. Also note, if only one column is specified in the select clause of this query, the value for this column is used for both display and return purposes.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(3,job,'SELECT DISTINCT job FROM
emp') job
FROM emp
```

35.18 SWITCH Function

This function dynamically generates flip toggle item. If On/Off value and label are not passed, it renders Yes/No toggle. This function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SWITCH (
    p_idx          IN NUMBER,
    p_value        IN VARCHAR2,
    p_on_value     IN VARCHAR2 DEFAULT 'Y',
    p_on_label     IN VARCHAR2 DEFAULT 'Yes',
    p_off_value    IN VARCHAR2 DEFAULT 'N',
    p_off_label    IN VARCHAR2 DEFAULT 'No',
    p_item_id      IN VARCHAR2 DEFAULT NULL,
    p_item_label   IN VARCHAR2,
    p_attributes   IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.
p_value	Form element current value.
p_on_value	The value of the item if the user picks On option.
p_on_label	The display text for the On option.
p_off_value	The value of the item if the user picks Off option.
p_off_label	The display text for the Off option.
p_item_id	HTML attribute ID for the <input> tag. Try concatenating some string with rownum to make it unique.
p_item_label	Invisible label created for the item.
p_attributes	Additional HTML attributes to use for the form item.

Example

The following example demonstrates the use of APEX_ITEM.SWITCH to generate a Yes/No flip toggle item where:

- A form array element F01 will be generated (p_idx parameter).
- The initial value for each element will be equal to N (p_value parameter).
- A HTML ID attribute will be generated for each row with the current rownum to uniquely identify. (p_item_id parameter). An ID of 'IS_MANAGER_2' is generated for row 2).
- A HTML label element will be generated for each row (p_item_label parameter).

```
SELECT
    ename "Name",
    APEX_ITEM.SWITCH (
        p_idx => 1,
        p_value => 'N',
        p_item_id => 'IS_MANAGER_' || rownum,
        p_item_label => apex_escape.html(ename) || ': Is Manager'
    ) "Is Manager"
FROM emp;
```

35.19 TEXT Function

This function generates text fields (or text input form items) from a SQL query.

Syntax

```
APEX_ITEM.TEXT(
    p_idx      IN NUMBER,
    p_value    IN VARCHAR2 DEFAULT NULL,
    p_size     IN NUMBER   DEFAULT NULL,
    p_maxlength IN NUMBER  DEFAULT NULL,
    p_attributes IN VARCHAR2 DEFAULT NULL,
    p_item_id  IN VARCHAR2 DEFAULT NULL,
```

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

Parameters

Parameter	Description
p_idx	Number to identify the item you want to generate. The number determines which G_FXX global is populated. See also APEX_APPLICATION .
p_value	Value of a text field item.
p_size	Controls HTML tag attributes (such as disabled).
p_maxlength	Maximum number of characters that can be entered in the text box.
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.

Example

The following sample query demonstrates how to generate one update field for each row. Note that the ename, sal, and comm columns use the APEX_ITEM.TEXT function to generate an HTML text field for each row. Note also that each item in the query is passed a unique p_idx parameter to ensure that each column is stored in its own array.

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno) ||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

35.20 TEXTAREA Function

This function creates text areas.

Syntax

```
APEX_ITEM.TEXTAREA (
    p_idx      IN      NUMBER,
    p_value    IN      VARCHAR2 DEFAULT NULL,
    p_rows     IN      NUMBER DEFAULT 40,
    p_cols     IN      NUMBER DEFAULT 4,
    p_attributes IN      VARCHAR2 DEFAULT NULL,
    p_item_id  IN      VARCHAR2 DEFAULT NULL,
```

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

Parameters

Parameter	Description
p_idx	Number to identify the item you want to generate. The number determines which G_FXX global is populated. See also APEX_APPLICATION .
p_value	Value of the text area item.
p_rows	Height of the text area (HTML rows attribute).
p_cols	Width of the text area (HTML column attribute).
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <textarea> tag.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to create a text area based on a SQL query.

```
SELECT APEX_ITEM.TEXTAREA(3,ename,5,80) a
FROM emp
```

35.21 TEXT_FROM_LOV Function

Displays an item as text, deriving the display value of the named LOV.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV (
    p_value      IN      VARCHAR2 DEFAULT NULL,
    p_lov        IN      VARCHAR2,
    p_null_text  IN      VARCHAR2 DEFAULT '%' )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	Value of a field item. Note that if p_value is not located in the list of values, p_null_text is displayed.
p_lov	Text name of a shared list of values. This list of values must be defined in your application.
p_null_text	Value displayed when the value of the field item is NULL.

Example

The following example demonstrates how to derive the display value from a named LOV (EMPNO_ENAME_LOV).

```
SELECT APEX_ITEM.TEXT_FROM_LOV(empno, 'EMPNO_ENAME_LOV') c FROM emp
```

35.22 TEXT_FROM_LOV_QUERY Function

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

Parameter	Description
p_value	Value of a field item.
p_query	SQL query that is expected to select two columns, a display column and a return column. For example: SELECT dname, deptno FROM dept
p_null_text	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. Value to be displayed when the value of the field item is NULL or a corresponding entry is not located for the value p_value in the list of values query.

Example

The following example demonstrates how to derive the display value from a query.

```
SELECT APEX_ITEM.TEXT_FROM_LOV_QUERY(empno, 'SELECT ename, empno FROM emp') c
from emp
```

APEX_JAVASCRIPT

The APEX_JAVASCRIPT package provides utility functions for adding dynamic JavaScript code to HTTP output. This package is usually used for plug-in development.

- [ADD_3RD_PARTY_LIBRARY_FILE Procedure \(Deprecated\)](#)
- [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_JET 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](#)

36.1 ADD_3RD_PARTY_LIBRARY_FILE Procedure (Deprecated)

 **Caution:**

This API is deprecated and will be removed in a future release.

This procedure adds the script tag to load a third-party JavaScript library file and also takes into account the specified CDN (content delivery network) for the application.

Supported libraries include:

- jQuery
- jQueryUI

Syntax

```
APEX_JAVASCRIPT.ADD_3RD_PARTY_LIBRARY_FILE (
    p_library      IN VARCHAR2,
    p_file_name    IN VARCHAR2 DEFAULT NULL,
    p_directory    IN VARCHAR2 DEFAULT NULL,
    p_version      IN VARCHAR2 DEFAULT NULL,
    p_attributes   IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameters	Description
p_library	Use one of the <code>c_library_*</code> constants.
p_file_name	Specifies the file name excluding version, .min, and .js.
p_directory	(Optional) Directory where the file <code>p_file_name</code> is located.
p_version	(Optional) If no value is provided, then uses the same version shipped with APEX.
p_attributes	Extra attributes to add to the script tag.

 **Note:**

Callers are responsible for escaping this parameter.

Example

This example loads the JavaScript file of the Draggable feature of jQuery UI.

```
apex_javascript.add_3rd_party_library_file (
    p_library    => apex_javascript.c_library_jquery_ui,
    p_file_name  => 'jquery.ui.draggable' )
```

36.2 ADD_ATTRIBUTE Function Signature 1

This function returns the attribute and the attribute's escaped text surrounded by double quotation marks.

 **Note:**

This function does not escape HTML tags. It only prevents HTML tags from breaking the JavaScript object attribute assignment. To prevent XSS (cross site scripting) attacks, you must also call `SYS.HTF.ESCAPE_SC` to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name      IN VARCHAR2,
    p_value     IN VARCHAR2,
    p_omit_null IN BOOLEAN:=TRUE,
    p_add_comma IN BOOLEAN:=TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Text to be assigned to the JavaScript object attribute.
p_omit_null	If p_omit_null is TRUE and p_value is NULL, the function returns nothing. If p_omit_null is FALSE and p_value is NULL, the value null is returned (for example, test:null).
p_add_comma	If set to TRUE, a trailing comma is added when a value is returned.

Example

Adds a call to the addEmployee JavaScript function and passes in a JavaScript object with different attribute values. The output of this call looks like:

```
addEmployee(
    {"FirstName":"John",
     "LastName":"Doe",
     "Salary":2531.29,
     "Birthday":new Date(1970,1,15,0,0,0),
     "isSalesman":true
});
```

As the last attribute you should use the parameter combination FALSE (p_omit_null), FALSE (p_add_comma) so that the last attribute is always generated. This avoids that you have to check for the other parameters if a trailing comma should be added or not.

```
apex_javascript.add_onload_code (
    'addEmployee(' ||
        '{' ||
            apex_javascript.add_attribute('FirstName',
sys.htf.escape_sc(l_first_name)) ||
            apex_javascript.add_attribute('LastName',
sys.htf.escape_sc(l_last_name)) ||
                apex_javascript.add_attribute('Salary',      l_salary) ||
                apex_javascript.add_attribute('Birthday',    l_birthday) ||
                apex_javascript.add_attribute('isSalesman', l_is_salesman, false,
false) ||
        '}');' );
```

36.3 ADD_ATTRIBUTE Function Signature 2

This function returns the attribute and the attribute's number.

Syntax

```
APEX_JAVASCRIPT.ADD_ATTRIBUTE (
    p_name      IN VARCHAR2,
    p_value     IN NUMBER,
    p.omit_null IN BOOLEAN:=TRUE,
    p.add_comma IN BOOLEAN:=TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Number which should be assigned to the JavaScript object attribute.
p.omit_null	If p.omit_null is TRUE and p_value is NULL, the function returns nothing. If p.omit_null is FALSE and p_value is NULL, the value null is returned (for example, test:null).
p.add_comma	If set to TRUE, a trailing comma is added when a value is returned.

Example

See example for [ADD_ATTRIBUTE Function Signature 1](#).

36.4 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

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Boolean assigned to the JavaScript object attribute.
p.omit_null	If p.omit_null is TRUE and p_value is NULL, the function returns nothing. If p.omit_null is FALSE and p_value is NULL, the value null is returned (for example, test:null).
p.add_comma	If set to TRUE a trailing comma is added when a value is returned.

Example

See example for [ADD_ATTRIBUTE Function Signature 1](#).

36.5 ADD_ATTRIBUTE Function Signature 4

This function returns the attribute and the attribute's date.

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

Parameter	Description
p_name	Name of the JavaScript object attribute.
p_value	Date assigned to the JavaScript object attribute.
p_omit_null	If p_omit_null is TRUE and p_value is NULL the function returns nothing. If p_omit_null is FALSE and p_value is NULL, the value null is returned (for example, test:null).
p_add_comma	If set to TRUE a trailing comma is added when a value is returned.

Example

See example for [ADD_ATTRIBUTE Function Signature 1](#).

36.6 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

Parameter	Description
p_code	JavaScript code snippet. For example: \$s('P1_TEST',123);
p_key	Identifier for the code snippet. If specified and a code snippet with the same name has already been added, the new code snippet is ignored. If p_key is NULL the snippet is always added.

Example

The following example includes the JavaScript function initMySuperWidget in the HTML output. If the plug-in is used multiple times on the page and the add_inline_code is called multiple times, it is added once to the HTML output because all calls have the same value for p_key.

```
apex_javascript.add_inline_code (
    p_code => 'function initMySuperWidget() {' || chr(10) ||
        '    // do something'||chr(10)|| 
        '}';
    p_key  => 'my_super_widget_function' );
```

36.7 ADD_JET Procedure

This procedure adds the script tag to load the Oracle JET library.

Syntax

```
APEX_JAVASCRIPT.ADD_JET;
```

Example

The following example demonstrates how to only load the Oracle JET library if the widget isn't rendered as a native browser input field.

```
if l_display_as <> 'NATIVE' then
    apex_javascript.add_jet;
end if;
```

36.8 ADD_LIBRARY Procedure

This procedure adds the script tag to load a JavaScript library. If a library has been added, it is not added a second time.

Syntax

```
APEX_JAVASCRIPT.ADD_LIBRARY (
    p_name          IN VARCHAR2,
    p_directory     IN VARCHAR2,
    p_version       IN VARCHAR2 DEFAULT NULL,
    p_check_to_add_minified IN BOOLEAN DEFAULT FALSE,
    p_skip_extension IN BOOLEAN DEFAULT FALSE,
```

```

    p_ie_condition          IN VARCHAR2 DEFAULT NULL,
    p_requirejs_module     IN VARCHAR2 DEFAULT NULL,
    p_requirejs_js_expression IN VARCHAR2 DEFAULT NULL,
    p_requirejs_required   IN BOOLEAN DEFAULT FALSE,
    p_is_module             IN BOOLEAN DEFAULT FALSE,
    p_is_async               IN BOOLEAN DEFAULT FALSE,
    p_is_defer               IN BOOLEAN DEFAULT FALSE,
    p_attributes             IN VARCHAR2 DEFAULT NULL,
    p_key                    IN VARCHAR2 DEFAULT NULL )
  
```

Parameters

Parameter	Description
p_name	Name of the JavaScript file. Must not use .js when specifying.
p_directory	Directory where JavaScript library is loaded. Must have a trailing slash.
p_version	Version identifier.
p_check_to_add_minifie d	If TRUE, the procedure tests if it is appropriate to add .min extension and add it if appropriate. This is added if an application is not running in DEBUG mode, and omitted when in DEBUG mode.
p_skip_extension	If TRUE, the extension .js is NOT added.
p_ie_condition	Condition which is used as Internet Explorer condition.
p_requirejs_module	Module name which is used to expose the library to RequireJS.
p_requirejs_js_expression	JavaScript expression which is used to expose the library to the RequireJS module.
p_requirejs_required	This has to be true if the library uses RequireJS in its code to loading other JavaScript files.
p_key	If not specified, defaults to p_directory p_name p_version.
p_is_module	If true, adds type="module" to the script tag.
p_is_async	If true, adds attribute async to the script tag.
p_is_defer	If true adds attribute defer to the script tag. defer cannot be used in combination with async. defer should not be used in combination with type="module" as module scripts defer by default.
p_attributes	Extra attributes to add to the script tag.

 **Note:**

Callers are responsible for escaping this parameter.

Example

The following example includes the JavaScript library file named hammer-2.0.4.min.js (if the application has not been started from the Builder), or hammer-2.0.4.js (if the application has been started from the Builder or is running in DEBUG mode), from the directory specified by p_plugin.file_prefix. Since p_skip_extension is not specified, this defaults to .js. Also, since p_key is not specified, the key defaults to p_plugin.file_prefix||hammer-2.0.4.

Hammer is a JavaScript library which exposes itself to RequireJS using `hammerjs` as module name.

```
apex_javascript.add_library (
    p_name                => 'hammer-2.0.4#MIN#',
    p_directory           => p_plugin.file_prefix,
    p_requirejs_module   => 'hammerjs',
    p_requirejs_js_expression => 'Hammer' );
```

36.9 ADD_REQUIREJS Procedure

This procedure adds the script tag to load the RequireJS library.

Syntax

```
APEX_JAVASCRIPT.ADD_REQUIREJS;
```

36.10 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

Parameter	Description
<code>p_module</code>	
<code>p_js_expression</code>	

Example

```
apex_javascript.add_requirejs_define (
    p_module      => 'hammerjs',
    p_js_expression => 'Hammer' );
```

36.11 ADD_ONLOAD_CODE Procedure

This procedure adds a JavaScript code snippet to the HTML output which the onload event executes. 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

Parameter	Description
p_code	JavaScript code snippet to execute during the onload event.
p_key	Any name to identify the specified code snippet. If specified, the code snippet is added if there has been no other call with the same p_key. If p_key is NULL the code snippet is always added.

Example

Adds the JavaScript call `initMySuperWidget()` to the onload buffer. If the plug-in is used multiple times on the page and the `add_onload_code` is called multiple times, it is added once to the HTML output because all calls have the same value for `p_key`.

```
apex_javascript.add_onload_code (
    p_code => 'initMySuperWidget();',
    p_key  => 'my_super_widget' );
```

36.12 ADD_VALUE Function Signature 1

This function returns the escaped text surrounded by double quotation marks. For example, this string could be returned "That\'s a test".

Note:

This function does not escape HTML tags. It only prevents HTML tags from breaking the JavaScript object attribute assignment. To prevent XSS (cross site scripting) attacks, you must also call `SYS.HTF.ESCAPE_SC` to prevent embedded JavaScript code from being executed when you inject the string into the HTML page.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value        IN VARCHAR2,
    p_add_comma   IN BOOLEAN :=TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	Text to be escaped and wrapped by double quotation marks.

Parameter	Description
p_add_comma	If p_add_comma is TRUE a trailing comma is added.

Example

This example adds some JavaScript code to the onload buffer. The value of p_item.attribute_01 is first escaped with htf.escape_sc to prevent XSS attacks and then assigned to the JavaScript variable lTest by calling apex_javascript.add_value. Add_value takes care of properly escaping the value and wrapping it with double quotation marks. Because commas are not wanted, p_add_comma is set to FALSE.

```
apex_javascript.add_onload_code (
    'var lTest = ''||
apex_javascript.add_value(sys.htf.escape_sc(p_item.attribute_01),
FALSE)||';'||chr(10)|||
    'showMessage(lTest);' );
```

36.13 ADD_VALUE Function Signature 2

This function returns p_value as JavaScript number, if p_value is NULL the value null is returned.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN NUMBER,
    p_add_comma     IN BOOLEAN :=TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	Number which should be returned as JavaScript number.
p_add_comma	If p_add_comma is TRUE a trailing comma is added. Default is TRUE.

Example

See example for [ADD_VALUE Function Signature 1](#).

36.14 ADD_VALUE Function Signature 3

This function returns p_value as JavaScript boolean. If p_value is NULL the value null is returned.

Syntax

```
APEX_JAVASCRIPT.ADD_VALUE (
    p_value          IN BOOLEAN,
    p_add_comma     IN BOOLEAN :=TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	Boolean which should be returned as JavaScript boolean.
p_add_comma	If p_add_comma is TRUE a trailing comma is added. Default is TRUE.

Example

See example for [ADD_VALUE Function Signature 1](#).

36.15 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 DATE,
    p_add_comma     IN BOOLEAN :=TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	Date which should be returned as JavaScript date object.
p_add_comma	If p_add_comma is TRUE a trailing comma is added. Default is TRUE.

Example

See example for [ADD_VALUE Function Signature 1](#).

36.16 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

Parameter	Description
p_text	Text to be escaped.

Example

Adds some JavaScript code to the onload buffer. The value of p_item.attribute_01 is first escaped with htf.escape_sc to prevent XSS attacks and then escaped with apex_javascript.escape to prevent special characters like a quotation mark from breaking the JavaScript code.

```
apex_javascript.add_onload_code (
    'var lTest = "' ||
    apex_javascript.escape(sys.htf.escape_sc(p_item.attribute_01)) || '"';
    || chr(10) ||
    'showMessage(lTest);' );
```

APEX_JSON

This package includes utilities that parse and generate JSON.

- APEX_JSON Overview and Examples
- Constants and Data Types
- CLOSE_ALL Procedure
- CLOSE_ARRAY Procedure
- CLOSE_OBJECT Procedure
- DOES_EXIST Function
- FIND_PATHS_LIKE Function
- FLUSH Procedure
- FREE_OUTPUT Procedure
- GET_BOOLEAN Function
- GET_CLOB Function
- GET_CLOB_OUTPUT Function
- GET_COUNT Function
- GET_DATE Function
- GET_MEMBERS Function
- GET_NUMBER Function
- GET_SDO_GEOMETRY Function
- GET_T_NUMBER Function
- GET_T_VARCHAR2 Function
- GET_VALUE Function
- GET_VALUE_KIND Function
- GET_VARCHAR2 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](#)
- [STRINGIFY Function Signature 5](#)
- [TO_MEMBER_NAME Function](#)
- [TO_XMLTYPE Function](#)
- [TO_XMLTYPE_SQL Function](#)
- [WRITE Procedure Signature 1](#)
- [WRITE Procedure Signature 2](#)
- [WRITE Procedure Signature 3](#)
- [WRITE Procedure Signature 4](#)
- [WRITE Procedure Signature 5](#)
- [WRITE Procedure Signature 6](#)
- [WRITE Procedure Signature 7](#)
- [WRITE Procedure Signature 8](#)
- [WRITE Procedure Signature 9](#)
- [WRITE Procedure Signature 10](#)
- [WRITE Procedure Signature 11](#)
- [WRITE Procedure Signature 12](#)
- [WRITE Procedure Signature 13](#)
- [WRITE Procedure Signature 14](#)
- [WRITE Procedure Signature 15](#)
- [WRITE Procedure Signature 16](#)
- [WRITE Procedure Signature 17](#)
- [WRITE Procedure Signature 18](#)
- [WRITE Procedure Signature 19](#)
- [WRITE Procedure Signature 20](#)
- [WRITE Procedure Signature 21](#)
- [WRITE_CONTEXT Procedure](#)

37.1 APEX_JSON Overview and Examples

To read from a string that contains JSON data, first use `parse()` to convert the string to an internal format. Then use the `get_%` routines (for example, `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 to the `SYS.HTP` buffer. To write to a temporary CLOB instead, use `initialize_clob_output()`, `get_clob_output()`, and `free_output()` for managing the output buffer.

Example 1

This example parses a JSON string and prints the value of member variable "a".

```
DECLARE
    s varchar2(32767) := '{ "a": 1, "b": ["hello", "world"] }';
BEGIN
    apex_json.parse(s);
    sys.dbms_output.put_line('a is '||apex_json.get_varchar2(p_path => 'a'));
END;
```

Example 2

This example converts a JSON string to XML and uses XMLTABLE to query member values.

```
select col1, col2
from xmltable (
    '/json/row'
    passing apex_json.to_xmlelement('["{"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;
```

37.2 Constants and Data Types

Parser Interface

The following constants are used for the parser interface:

```
subtype t_kind is binary_integer range 1 .. 8;
c_null      constant t_kind := 1;
c_true      constant t_kind := 2;
c_false     constant t_kind := 3;
c_number    constant t_kind := 4;
c_varchar2  constant t_kind := 5;
```

```
c_object    constant t_kind := 6;
c_array     constant t_kind := 7;
c_clob      constant t_kind := 8;
```

Storage for JSON Data

JSON data is stored in an index by varchar2 table. The JSON values are stored as records. The discriminator "kind" determines whether the value is null, true, false, a number, a varchar2, a clob, an object or an array. It depends on "kind" which record fields are used and how. If not explicitly mentioned below, the other record fields' values are undefined:

```
* c_null: -
* c_true: -
* c_false: -
* c_number: number_value contains the number value
* c_varchar2: varchar2_value contains the varchar2 value
* c_clob: clob_value contains the clob
* c_object: object_members contains the names of the object's members
* c_array: number_value contains the array length

type t_value is record (
    kind          t_kind,
    number_value  number,
    varchar2_value varchar2(32767),
    clob_value    clob,
    object_members apex_t_varchar2 );
type t_values is table of t_value index by varchar2(32767);
```

Default Format for Dates

```
c_date_iso8601 constant varchar2(30) := 'yyyy-mm-dd"T"hh24:mi:ss"Z"';
```

Default JSON Values Table

```
g_values t_values;
```

Errors Thrown for PARSE()

```
e_parse_error      exception;
pragma exception_init(e_parse_error, -20987);
```

37.3 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 [APEX_JSON Overview and Examples](#).

37.4 CLOSE_ARRAY Procedure

This procedure writes a close bracket symbol as follows:

```
]
```

Syntax

```
APEX_JSON CLOSE_ARRAY ();
```

Parameters

None.

Example

See [APEX_JSON Overview and Examples](#).

37.5 CLOSE_OBJECT Procedure

This procedure writes a close curly bracket symbol as follows:

```
}
```

Syntax

```
APEX_JSON CLOSE_OBJECT ();
```

Parameters

None.

Example

See [APEX_JSON Overview and Examples](#).

37.6 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

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns

Return	Description
TRUE	Given path points to an existing value.
FALSE	Given path does not point to an existing value

Example

This example parses a JSON string and prints whether it contains values under a path.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": true } ] }');
    IF apex_json.does_exist(p_path => 'items[%d].foo', p0 => 3, p_values => j)
    THEN
        dbms_output.put_line('found items[3].foo');
    END IF;
END;
```

37.7 FIND_PATHS_LIKE Function

This function returns paths into p_values that match a given pattern.

Syntax

```
APEX_JSON.FIND_PATHS_LIKE (
    p_return_path   IN VARCHAR2,
    p_subpath       IN VARCHAR2 DEFAULT NULL,
    p_value         IN VARCHAR2 DEFAULT NULL,
```

```
p_values          IN t_values DEFAULT g_values )
RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_return_path	Search pattern for the return path..
p_subpath	(Optional) Search pattern under p_return_path.
p_value	(Optional) Search pattern for value.
p_values	Parsed JSON members. Default is g_values.

Returns/Raised Errors

Return/Error	Description
apex_t_varchar2	Table of paths that match the pattern.
VALUE_ERROR	Raises this error if p_values (p_path is not an array or object.

Example

This example parses a JSON string, finds paths that match a pattern, and prints the values under the paths.

```
DECLARE
    j      apex_json.t_values;
    l_paths apex_t_varchar2;
BEGIN
    apex_json.parse(j, '{ "items": [ { "name": "Amulet of Yendor", "magical": true }, { "name": "Slippers", "magical": "rather not" } ] }');
    l_paths := apex_json.find_paths_like (
        p_values      => j,
        p_return_path => 'items[%]',
        p_subpath     => '.magical',
        p_value       => 'true' );
    dbms_output.put_line('Magical items:');
    FOR i in 1 .. l_paths.count LOOP
        dbms_output.put_line(apex_json.get_varchar2(p_values => j, p_path => l_paths(i)||'.name'));
    END LOOP;
END;
```

37.8 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;
```

37.9 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

```
APEX_JSON.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;
```

37.10 GET_BOOLEAN Function

This function returns a boolean member 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

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_default	The default value if the member does not exist.
p_values	Parsed JSON members. The default is g_values.

Returns

Return	Description
TRUE	Value at the given path position.
FALSE	Value at the given path position.
NULL	Value at the given path position.
VALUE_ERROR	Raises this error if p_values(p_path) is not boolean.

Example

This example parses a JSON string and prints the boolean value at a position.

```

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": true } ] }');
    IF apex_json.get_boolean(p_path=>'items[%d].foo', p0=>3,p_values=>j) THEN
        dbms_output.put_line('items[3].foo is true');
    END IF;
END;

```

37.11 GET_CLOB Function

This function returns clob member value. This function auto-converts varchar2, boolean, and number values.

Syntax

```

GET_CLOB (
    p_path      IN VARCHAR2,
    p0          IN VARCHAR2 DEFAULT NULL,
    p1          IN VARCHAR2 DEFAULT NULL,
    p2          IN VARCHAR2 DEFAULT NULL,
    p3          IN VARCHAR2 DEFAULT NULL,
    p4          IN VARCHAR2 DEFAULT NULL,
    p_default   IN CLOB      DEFAULT NULL,

```

```
p_values  in t_values DEFAULT g_values )
RETURN CLOB
```

Parameters

Parameter	Description
p_values	Parsed JSON members. defaults to g_values.
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced with pN and every i-th %s or %d is replaced with p[i-1].
p_default	Default value if the member does not exist.

Returns/Raised Errors

Return/Error	Description
a_clob	Value at the given path position.
VALUE_ERROR	If p_values(p_path) is an array or an object.

Example

Parse a JSON string and print the value at a position.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_clob (
        p_values => j,
        p_path => 'items[%d].foo',
        p0 => 3));
END;
```

37.12 GET_CLOB_OUTPUT Function

Returns the temporary CLOB that you created with INITIALIZE_CLOB_OUTPUT.

Syntax

```
APEX_JSON.GET_CLOB_OUTPUT (
    p_free  IN BOOLEAN  DEFAULT FALSE )
RETURN CLOB;
```

Parameters

Parameter	Description
p_free	If true, frees output resources. Defaults to false.

Example 1

This example configures APEX_JSON for CLOB output, generates JSON, prints the CLOB with DBMS_OUTPUT, and finally frees 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;
```

Example 2

This example configures APEX_JSON for CLOB output, generates JSON, and prints and frees the CLOB with DBMS_OUTPUT and GET_CLOB_OUTPUT.

```
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( p_free => true ) );
END;
```

37.13 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

Parameter	Description
p_path	Index into p_values.

Parameter	Description
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Return/Error	Description
NUMBER	The number of array elements or object members or null if the array or object could not be found
VALUE_ERROR	Raises this error if p_values (p_path) is not an array or object.

Example

This example parses a JSON string and prints the number of members at positions.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    dbms_output.put_line(apex_json.get_count(p_path=>'.',p_values=>j)); -- 2
    (foo and bar)
    dbms_output.put_line(apex_json.get_count(p_path=>'bar',p_values=>j)); -- 4
END;
```

37.14 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 NULL,
    p_values        IN t_values DEFAULT g_values )
RETURN DATE;
```

Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].

Parameter	Description
p_default	The default value if the member does not exist.
p_format	The date format mask.
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Return	Description
DATE	Returns the date.
VALUE_ERROR	Raises this error if p_values(p_path) is not a date.

Example

This example parses a JSON string and prints the value at a position.

```

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo":'
"2014-04-29T10:08:00Z" }] }');

    dbms_output.put_line(to_char(apex_json.get_date(p_path=>'items[%d].foo',p0=>3,
        p_values=>j), 'DD-Mon-YYYY'));
END;

```

37.15 GET_MEMBERS Function

This function returns the table of OBJECT_MEMBERS names for an object.

Syntax

```

APEX_JSON.GET_MEMBERS (
    p_path          IN VARCHAR2,
    p0              IN VARCHAR2 DEFAULT NULL,
    p1              IN VARCHAR2 DEFAULT NULL,
    p2              IN VARCHAR2 DEFAULT NULL,
    p3              IN VARCHAR2 DEFAULT NULL,
    p4              IN VARCHAR2 DEFAULT NULL,
    p_values        IN t_values DEFAULT g_values )
RETURN APEX_T_VARCHAR2;

```

Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th% or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Return/Error	Description
OBJECT_MEMBERS	The OBJECT_MEMBERS of the object or null if the object could not be found.
VALUE_ERROR	Raises this error if p_values(p_path) is not an array or object.

Example

This example parses a JSON string and prints members at positions.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    dbms_output.put_line(apex_json.get_members(p_path=>'.',p_values=>j)(1));
-- foo
    dbms_output.put_line(apex_json.get_members(p_path=>'.',p_values=>j)(2));
-- bar
END;
```

37.16 GET_NUMBER Function

This function returns a numeric member 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

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th% or %d is replaced by the p[i-1].
p_default	The default value if the member does not exist.
p_values	Parsed JSON members. The default is g_values.

Returns and Raised Errors

Return	Description
NUMBER	The value at the given path position.
VALUE_ERROR	Raises this error if p_values(p_path) is not a number.

Example

This example parses a JSON string and prints the value at a position.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_number(p_path=>'items[%d].foo',p0=>
3,p_values=>j));
END;
```

37.17 GET_SDO_GEOMETRY Function

This function returns SDO_GEOMETRY member value from a GeoJSON member. This function supports only two-dimensional geometry objects.

 **Note:**

This function is **only** available if SDO_GEOMETRY (Oracle Locator) is installed in the database.

Syntax

```
APEX_JSON.GET_SDO_GEOMETRY FUNCTION (
    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_srid          IN NUMBER    DEFAULT 4326,
    p_values        IN t_values DEFAULT g_values )
RETURN mdsys.sdo_geometry;
```

Parameters

Parameter	Description
p_values	Parsed JSON members. Defaults to g_values.
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].

Parameter	Description
p_srid	Coordinate system (SRID) to return the SDO_GEOOMETRY in.

Returns

Return	Description
a geometry	Value at the given path position.

Errors Raised

Raise	Description
VALUE_ERROR	If p_values(p_path) is not a GeoJSON object.

Example

The following example parses a JSON string and prints the value at a position.

```

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "geom":
{"type":"Point","coordinates":[-122.7783356,38.8198318,1.85 ] } } ] }');
    dbms_output.put_line(to_char(apex_json.get_sdo_geometry (
        p_values => j,
        p_path   => 'items[%d].geom',
        p0       => 3) ) );
END;

```

37.18 GET_T_NUMBER Function

This function returns the numeric attributes of an array.

Syntax

```

APEX_JSON.GET_T_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_values    IN t_values DEFAULT q_values )
RETURN apex_t_number;

```

Parameters

Parameter	Description
p_path	Index into p_values.

Parameter	Description
p[0-4]	Each %N in p_path is replaced by pN and every i-th% or %d is replaced by the p[i-1].
p_values	Parsed JSON members. Default p_values.

Returns

Array member values if the referenced t_value is an array. An array with just the referenced value if its type can be converted to a number.

Return	Description
VALUE_ERROR	On conversion errors.

Example

This example parses a JSON string and prints the value at position 1.

```

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
2:222
1:333

```

37.19 GET_T_VARCHAR2 Function

This function returns the varchar2 attributes of an array.

Syntax

```

APEX_JSON.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 apex_t_varchar2;

```

Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th% or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns

Array member values if the referenced t_value is an array. An array with just the referenced value if its type can be converted to a varchar2.

Raises

Return	Description
VALUE_ERROR	On conversion errors.

Example

This example parses a JSON and prints the value at position 1.

```

declare
    j          apex_json.t_values;
    l_elements apex_t_varchar2;
begin
    apex_json.parse(j, '{ "foo": ["one", "two"], "bar": "three" }');
    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

```

37.20 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

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th% or %d is replaced by the p[i-1].
p_values	Parsed JSON members. The default is g_values.

Returns/Raised Errors

Return	Description
t_value	The t_value at the given path position. The record attributes are null if no data is found.
VALUE_ERROR	Raises this error if p_values (p_path) is not an array or object.

Example

This example parses a JSON string and prints attributes of values at positions.

```
DECLARE
    j apex_json.t_values;
    v apex_json.t_value;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    v := apex_json.get_value(p_path=>'bar[%d]',p0=> 2,p_values=>j); --
returns the t_value for bar[2]
    dbms_output.put_line(v.number_value); -- 2
    v := apex_json.get_value(p_path=>'does.not.exist',p_values=>j);
    dbms_output.put_line(case when v.kind is null then 'not found!' end);
END;
```

37.21 GET_VALUE_KIND Function

This function returns the kind of the value at a path position.

Syntax

```
APEX_JSON.GET_VALUE_KIND (
    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_kind;
```

Parameters

Parameter	Description
p_values	Parsed JSON members. Defaults to g_values.
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].

Table 37-1 Returns

Return	Description
t_kind	The t_kind of the value at the given path position. Returns NULL if no data is found.

Example

This example parses a JSON string and prints the kind of an attribute.

```
DECLARE
    j apex_json.t_values;
    k apex_json.t_kind;

    PROCEDURE print_kind( p_kind in apex_json.t_kind ) IS
    BEGIN
        dbms_output.put_line(
            CASE p_kind
                WHEN apex_json.c_null      THEN 'NULL'
                WHEN apex_json.c_true      THEN 'true'
                WHEN apex_json.c_false     THEN 'false'
                WHEN apex_json.c_number    THEN 'NUMBER'
                WHEN apex_json.c_varchar2  THEN 'VARCHAR2'
                WHEN apex_json.c_object    THEN 'OBJECT'
                WHEN apex_json.c_array     THEN 'ARRAY'
                WHEN apex_json.c_clob      THEN 'CLOB' end );
    END print_kind;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": [1, 2, 3, 4] }');
    k := apex_json.get_value_kind (
        p_values => j,
        p_path   => 'bar[%d]',
```

```

        p0      => 2); -- returns the t_value for bar[2]
print_kind(k);      -- 'NUMBER'
k := apex_json.get_value_kind (
    p_values => j,
    p_path   => 'bar');
print_kind(k);      -- 'ARRAY'
END;

```

37.22 GET_VARCHAR2 Function

This function returns a varchar2 member value. This function converts boolean and number values to varchar2 values.

Syntax

```

APEX_JSON.GET_VARCHAR2 (
    p_path          IN VARCHAR2,
    p0              IN VARCHAR2 DEFAULT NULL,
    p1              IN VARCHAR2 DEFAULT NULL,
    p2              IN VARCHAR2 DEFAULT NULL,
    p3              IN VARCHAR2 DEFAULT NULL,
    p4              IN VARCHAR2 DEFAULT NULL,
    p_default       IN BOOLEAN  DEFAULT NULL,
    p_values        IN t_values DEFAULT g_values )
RETURN VARCHAR2;

```

Parameters

Parameter	Description
p_path	Index into p_values.
p[0-4]	Each %N in p_path is replaced by pN and every i-th %s or %d is replaced by the p[i-1].
p_default	The default value if the member does not exist.
p_values	Parsed JSON members. The default is g_values.

Returns and Raised Errors

Return/Error	Description
VARCHAR2	This is the value at the given path position.
VALUE_ERROR	Raises this error if p_values(p_path) is not an array or object.

Example

This example parses a JSON string and prints the value at a position.

```

DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "items": [ 1, 2, { "foo": 42 } ] }');
    dbms_output.put_line(apex_json.get_varchar2(p_path=>'items[%d].foo',p0=>

```

```
3,p_values=>j));
END;
```

37.23 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, call FREE_OUTPUT() at the end to free the CLOB.

Syntax

```
APEX_JSON.INITIALIZE_CLOB_OUTPUT (
    p_dur          IN PLS_INTEGER DEFAULT sys.dbms_lob.call,
    p_cache        IN BOOLEAN      DEFAULT TRUE,
    p_indent       IN PLS_INTEGER DEFAULT NULL,
    p_preserve     IN BOOLEAN      DEFAULT FALSE )
```

Parameters

Parameter	Description
p_dur	Duration of the temporary CLOB. This can be DBMS_LOB.SESSION or DBMS_LOB.CALL (the default).
p_cache	Specifies if the lob should be read into buffer cache or not.
p_indent	Indent level. Defaults to 2 if debug is turned on, 0 otherwise.
p_preserve	Whether to preserve the currently active output object. After calling FREE_OUTPUT, subsequent write calls will be executed on the preserved output. Defaults to FALSE. If HTP output has already been initialized and a CLOB needs to be created, use p_preserve => true. After FREE_OUTPUT, subsequent output will be directed to the original HTP output again. If p_preserve is true, you must call FREE_OUTPUT after JSON processing.

Example

This example configures APEX_JSON for CLOB output, generates JSON, prints the CLOB with DBMS_OUTPUT, and finally frees the CLOB.

```
BEGIN
    apex_json.initialize_clob_output( p_preserve => true );

    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;
```

37.24 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

Parameter	Description
p_http_header	If TRUE (default), writes an application/JSON mime type header.
p_http_cache	This parameter is only relevant if p_http_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.
http_cache_etag	If not null, writes an etag header. This parameter is only used if P_HTTP_CACHE is true.
p_indent	Indent level. Defaults to 2, if debug is turned on, otherwise defaults to 0.

Example

This example configures APEX_JSON to not emit default headers, because they are written directly.

```
BEGIN
    apex_json.initialize_output (
        p_http_header => false );

    sys.owa_util.mime_header('application/json', false);
    sys.owa_util.status_line(429, 'Too Many Requests');
    sys.owa_util.http_header_close;
    --
    apex_json.open_object;
    apex_json.write('maxRequestsPerSecond', 10);
    apex_json.close_object;
END;
```

37.25 OPEN_ARRAY Procedure

This procedure writes an open bracket symbol as follows:

[

Syntax

```
APEX_JSON.OPEN_ARRAY (
    p_name      IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_name	If not null, write an object attribute name and colon before the opening bracket.

Example

This example performs a `write { "array": [1 , []] }.`

```
BEGIN
    apex_json.open_object; -- {
    apex_json.open_array('array'); -- "array": [
    apex_json.write(1); -- 1
    apex_json.open_array; -- ,
    apex_json.close_array; -- ]
    apex_json.close_array; -- ]
    apex_json.close_object; -- }
END;
```

37.26 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

Parameter	Description
p_name	If not null, write an object attribute name and colon before the opening brace.

Example

This example performs a `write { "obj": { "obj-attr": "value" } }.`

```
BEGIN
    apex_json.open_object; -- {
    apex_json.open_object('obj'); -- "obj": {
    apex_json.write('obj-attr', 'value'); -- "obj-attr": "value"
```

```
    apex_json.close_all; -- } }
END;
```

37.27 PARSE Procedure Signature 1

This procedure parses a JSON-formatted VARCHAR2 or CLOB and puts the members into p_values.

Syntax

```
APEX_JSONPARSE (
    p_values    IN OUT NOCOPY    t_values,
    p_source    IN VARCHAR2,
    p_strict    IN BOOLEAN        DEFAULT TRUE );

APEX_JSONPARSE (
    p_values    IN OUT NOCOPY    t_values,
    p_source    IN CLOB,
    p_strict    IN BOOLEAN        DEFAULT TRUE );
```

Parameters

Parameter	Description
p_values	An index by VARCHAR2 result array which contains the JSON members and values. The default is g_values.
p_source	The JSON source (VARCHAR2 or CLOB)
p_strict	If TRUE (default), enforce strict JSON rules

Example

This example parses JSON and prints member values.

```
DECLARE
    l_values apex_json.t_values;
BEGIN
    apex_json.parse (
        p_values => l_values,
        p_source => '{ "type": "circle", "coord": [10, 20] }' );
    sys.htp.p('Point at ' ||
        apex_json.get_number (
            p_values => l_values,
            p_path   => 'coord[1]' ) ||
        ', ' ||
        apex_json.get_number (
            p_values => l_values,
            p_path   => 'coord[2]' ));
END;
```

37.28 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_JSONPARSE (
    p_source      IN VARCHAR2,
    p_strict      IN BOOLEAN DEFAULT TRUE );

APEX_JSONPARSE (
    p_source      IN CLOB,
    p_strict      IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
<code>p_source</code>	The JSON source (<code>VARCHAR2</code> or <code>CLOB</code>).
<code>p_strict</code>	If <code>TRUE</code> (default), enforce strict JSON rules.

Example

This example parses JSON and prints member values.

```
apex_json.parse('{ "type": "circle", "coord": [10, 20] }');
sys.htp.p('Point at ' ||
          apex_json.get_number(p_path=>'coord[1]'') ||
          ', ' ||
          apex_json.get_number(p_path=>'coord[2]''));
```

37.29 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

Parameter	Description
<code>p_value</code>	The string to be converted.

Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example is a query that returns a JSON varchar2 value.

```
select apex_json.stringify('line 1'||chr(10)||'line 2') from dual;
```

37.30 STRINGIFY Function Signature 2

This function converts a number to an escaped JSON value.

Syntax

```
APEX_JSON.STRINGIFY (
    p_value  IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	The number to be converted.

Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example is a query that returns a JSON number value.

```
select apex_json.stringify(-1/10) from dual
```

37.31 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

Parameter	Description
p_value	The date value to be converted.

Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example is a query that returns a JSON varchar2 value that is suitable to be converted to dates.

```
select apex_json.stringify(sysdate) from dual
```

37.32 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

Parameter	Description
p_value	The boolean value to be converted.

Returns

Return	Description
VARCHAR2	The converted and escaped JSON value.

Example

This example demonstrates printing JSON boolean values.

```
BEGIN
    sys.htp.p(apex_json.stringify(true));
    sys.htp.p(apex_json.stringify(false));
END;
```

37.33 STRINGIFY Function Signature 5

This function converts `p_value` to a GeoJSON value.

 **Note:**

This signature is **only** available if SDO_GEOOMETRY (Oracle Locator) is installed in the database.

Syntax

```
APEX_JSON.STRINGIFY (
    p_value IN mdsys.sdo_geometry )
RETURN CLOB;
```

Parameters

Parameter	Description
<code>p_value</code>	The <code>sdo_geometry</code> value to be converted.

Returns

Return	Description
<code>CLOB</code>	The GeoJSON value.

Example

The following example prints GeoJSON values.

```
BEGIN
    sys.hpt.p(apex_json.stringify(
        mdsys.sdo_geometry( 2001, 4326, sdo_point_type( 10, 50,
null ), null, null ) ) );
END;
```

37.34 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

Parameter	Description
p_string	The raw member name.

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;
```

Output:

```
Unquoted: member_name
Quoted:    "Hello\"World"
```

37.35 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

Parameter	Description
p_source	The JSON source (<code>VARCHAR2</code> or <code>CLOB</code>)
p_strict	If <code>TRUE</code> (default), enforce strict JSON rules

Returns

Return	Description
sys.xmltype	An xmltype representation of the JSON data.

Example

This example parses JSON and prints the XML representation.

```
DECLARE
    l_xml xmltype;
BEGIN
    l_xml := apex_json.to_xmltype('{ "items": [ 1, 2, { "foo": true } ] }');
    dbms_output.put_line(l_xml.getstringval);
END;
```

37.36 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

```
APEX_JSON.TO_XMLTYPE_SQL (
    p_source    IN VARCHAR2,
    p_strict    IN VARCHAR2 DEFAULT 'Y' )
RETURN sys.xmltype;

APEX_JSON.TO_XMLTYPE_SQL (
    p_source    IN CLOB,
    p_strict    IN VARCHAR2 DEFAULT 'Y' )
RETURN sys.xmltype;
```

Parameters

Parameter	Description
p_source	The JSON source (VARCHAR2 or CLOB)
p_strict	If Y (default), enforce strict JSON rules

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.

```
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()'
    );
```

37.37 WRITE Procedure Signature 1

This procedure writes an array attribute of type VARCHAR2.

Syntax

```
APEX_JSON.WRITE (
    p_value      IN VARCHAR2 );
```

Parameters

Parameter	Description
p_value	The value to be written.

Example

This example writes an array containing 1, "two", "long text", false, the current date and a JSON representation of an XML document.

```
DECLARE
    l_clob clob := 'long text';
    l_xml sys.xmltype := sys.xmltype('<obj><foo>1</foo><bar>2</bar></obj>');
BEGIN
    -- if not executed within an APEX session context, JSON output needs to be
    initialized first
    apex_json.initialize_clob_output;
    apex_json.open_object;
    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; -- ]
apex_json.close_object;
dbms_output.put_line(apex_json.get_clob_output);
apex_json.free_output;
END;
```

37.38 WRITE Procedure Signature 2

This procedure writes an array attribute of type `clob`.

Syntax

```
APEX_JSON.WRITE (
    p_value      IN CLOB );
```

Parameters

Parameter	Description
<code>p_value</code>	The value to be written.

Example

See [WRITE Procedure Signature 1](#).

37.39 WRITE Procedure Signature 3

This procedure writes an array attribute of type `NUMBER`.

Syntax

```
APEX_JSON.WRITE (
    p_value      IN NUMBER );
```

Parameters

Parameter	Description
<code>p_value</code>	The value to be written.

Example

See [WRITE Procedure Signature 1](#).

37.40 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

Parameter	Description
p_value	The value to be written.
p_format	The date format mask (default c_date_iso8601).

Example

See [WRITE Procedure Signature 1](#).

37.41 WRITE Procedure Signature 5

This procedure writes an array attribute of type boolean.

Syntax

```
APEX_JSON.WRITE (
    p_value      IN BOOLEAN );
```

Parameters

Parameter	Description
p_value	The value to be written.

Example

See [WRITE Procedure Signature 1](#).

37.42 WRITE Procedure Signature 6

This procedure writes an array attribute of type sys.xmlelement. 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

Parameter	Description
p_value	The value to be written.

Example

See [WRITE Procedure Signature 1](#).

37.43 WRITE Procedure Signature 7

This procedure writes an array with all rows that the cursor returns. Each row is a separate object. If the query contains object type, collection, or cursor columns, the procedure uses `write(xmltype)` to generate JSON. Otherwise, it uses `DBMS_SQL` to fetch rows and the `write()` procedures for the appropriate column data types for output. If the column type is `varchar2` and the uppercase value is '`TRUE`' or '`FALSE`', it generates boolean values.

Syntax

```
APEX_JSON.WRITE (
    p_cursor      IN OUT NOCOPY sys_refcursor );
```

Parameters

Parameter	Description
p_cursor	The cursor.

Example

This example writes an array containing JSON objects for departments 10 and 20.

```
DECLARE
    c sys_refcursor;
BEGIN
    open c for select deptno, dname, loc from dept where deptno in (10, 20);
    apex_json.write(c);
END;
```

This is the output:

```
[ { "DEPTNO":10 , "DNAME":"ACCOUNTING" , "LOC":"NEW YORK" }
, { "DEPTNO":20 , "DNAME":"RESEARCH" , "LOC":"DALLAS" } ]
```

37.44 WRITE Procedure Signature 8

This procedure writes array attribute of type SDO_GEOMETRY.

 **Note:**

This signature is **only** available if SDO_GEOMETRY (Oracle Locator) is installed in the database.

Syntax

```
APEX_JSON.WRITE (
    p_value      IN mdsys.sdo_geometry );
```

Parameters

Parameter	Description
p_value	The value to be written.

37.45 WRITE Procedure Signature 9

This procedure writes an object attribute of type VARCHAR2.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_value     IN VARCHAR2,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If TRUE, write NULL values. If FALSE (default), do not write NULLs.

Example

This example writes an object with named member attributes of various types. The comments to the right of the statements show the output that they generate.

```
DECLARE
    l_clob clob := 'long text';
    l_xml sys.xmltype := sys.xmltype('<obj><foo>1</foo><bar>2</bar></obj>');
BEGIN
    apex_json.open_object; -- {
    apex_json.write('a1', 1); -- "a1": 1
```

```

apex_json.write('a2', 'two'); -- , "a2": "two"
apex_json.write('a3', l_clob); -- , "a3": "long text"
apex_json.write('a4', false); -- , "a4": false
apex_json.write('a5', sysdate); -- , "a5": "2014-05-05T05:36:08Z"
apex_json.write('a6', l_xml); -- , "a6": { "foo": 1, "bar": 2 }
apex_json.close_object; -- }
END;

```

37.46 WRITE Procedure Signature 10

This procedure writes an object attribute of type CLOB.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_value     IN CLOB,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If TRUE, write NULL values. If FALSE (the default), do not write NULLs.

Example

See example for [WRITE Procedure Signature 9](#).

37.47 WRITE Procedure Signature 11

This procedure writes an object attribute of type NUMBER.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_value     IN NUMBER,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If TRUE, write NULL values. If false (the default), do not write NULLs.

Example

See example for WRITE Procedure Signature 9.

37.48 WRITE Procedure Signature 12

This procedure writes an object attribute of type date.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_value     IN DATE,
    p_format    IN VARCHAR2 DEFAULT c_date_iso8601,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_format	The date format mask (default apex_json.c_date_iso8601).
p_write_null	If TRUE, write NULL values. If FALSE (default), do not write NULL.

Example

See example for WRITE Procedure Signature 9.

37.49 WRITE Procedure Signature 13

This procedure writes an object attribute of type boolean.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_value     IN BOOLEAN,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If TRUE, write NULL values. If FALSE (default), do not write NULL.

Example

See example for WRITE Procedure Signature 9.

37.50 WRITE Procedure Signature 14

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 15](#). Otherwise, it uses `DBMS_SQL` to fetch rows and the `write()` procedures for the appropriate column data types for output. If the column type is `varchar2` and the uppercase value is 'TRUE' or 'FALSE', it generates boolean values.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_cursor    IN OUT NOCOPY sys_refcursor );
```

Parameters

Parameter	Description
<code>p_name</code>	The attribute name.
<code>p_cursor</code>	The cursor.

Example

This example writes an array containing JSON objects for departments 10 and 20, as an object member attribute.

```
DECLARE
    c sys_refcursor;
BEGIN
    open c for select deptno,
                    dname,
                    cursor(select empno,
                            ename
                           from emp e
                          where e.deptno=d.deptno) emps
                  from dept d;
    apex_json.open_object;
    apex_json.write('departments', c);
    apex_json.close_object;
END;

{
  "departments": [
    {"DEPTNO":10,
     "DNAME":"ACCOUNTING",
     "EMPS": [{"EMPNO":7839,"ENAME":"KING"}]},
    ...
    {"DEPTNO":40,"DNAME":"OPERATIONS","EMPS":null} ]
}
```

37.51 WRITE Procedure Signature 15

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

Parameter	Description
<code>p_name</code>	The attribute name.
<code>p_value</code>	The value to be written. The XML is converted to JSON
<code>p_write_null</code>	If <code>TRUE</code> , write <code>NULL</code> values. If <code>FALSE</code> (default), do not write <code>NULLS</code> .

Example

See example for [WRITE Procedure Signature 14](#).

37.52 WRITE Procedure Signature 16

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

Parameter	Description
<code>p_values</code>	The parsed JSON members.

Parameter	Description
p_path	The index into p_values.
p[0-4]	Each %N in p_path will be replaced by pN and every i-th %s or %d is replaced by p[i-1].

Example

This example parses a JSON string and writes parts of it.

```
DECLARE
    j apex_json.t_values;
BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": { "x": 1, "y": 2 } }');
    apex_json.write(j,'bar');
END;
```

37.53 WRITE Procedure Signature 17

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

Parameter	Description
p_name	The attribute name.
p_values	The parsed JSON members.
p_path	The index into p_values.
p[0-4]	Each %N in p_path will be replaced by pN and every i-th %s or %d is replaced by p[i-1].
p_write_null	If true, write NULL values. If false (the default), do not write NULLs.

Example

This example parses a JSON string and writes parts of it as an object member.

```
DECLARE
    j apex_json.t_values;
```

```

BEGIN
    apex_json.parse(j, '{ "foo": 3, "bar": { "x": 1, "y": 2 } }');
    apex_json.open_object; -- {
        apex_json.write('parsed-bar',j,'bar');-- "parsed-bar":{ "x":1 , "y":2 }
        apex_json.close_object; -- }
END;

```

37.54 WRITE Procedure Signature 18

This procedure writes an array attribute of type VARCHAR2.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_values    IN apex_t_varchar2,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_values	The VARCHAR2 array values to be written.
p_write_null	If true, write an empty array. If false (the default), do not write an empty array.

Example

This example writes an array containing a, b, c.

```

DECLARE
    l_values apex_t_varchar2 := apex_t_varchar2( 'a', 'b', 'c' );
BEGIN
    apex_json.open_object; -- {
    apex_json.write('array', l_values ); -- "array": [ "a", "b", "c" ]
    apex_json.close_object; -- }
END;
```

37.55 WRITE Procedure Signature 19

This procedure writes an array attribute of type NUMBER .

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_values    IN apex_t_number,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_values	The NUMBER array values to be written.
p_write_null	If true, write an empty array. If false (the default), do not write an empty array.

Example

This example writes an array containing 1, 2, 3.

```
DECLARE
    l_values apex_t_number := apex_t_number( 1, 2, 3 );
BEGIN
    apex_json.open_object;                      -- {
    apex_json.write('array', l_values );        --   "array": [ 1, 2, 3 ]
    apex_json.close_object;                     -- }
END;
```

37.56 WRITE Procedure Signature 20

This procedure writes a BLOB object attribute. The value will be Base64-encoded.

Syntax

```
APEX_JSON.WRITE (
    p_name      IN VARCHAR2
    p_value     IN BLOB,
    p_write_null IN BOOLEAN DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If TRUE, write an empty array. If FALSE (default), do not write an empty array.

Example

This example writes a JSON object with the a1, a2, a3, and a4 attributes. a3 is a BLOB, encoded in Base64 format.

```
DECLARE
    l_blob blob := to_blob( hextoraw('000102030405060708090a'));
BEGIN
    apex_json.open_object; -- {
    apex_json.write('a1', 1); -- "a1": 1
    apex_json.write('a2', 'two'); -- , "a2": "two"
    apex_json.write('a3', l_blob); -- , "a3": "AAECAwQFBgcICQo="

```

```

apex_json.write('a4', false); -- , "a4": false
apex_json.close_object; -- }
END;

```

37.57 WRITE Procedure Signature 21

This procedure writes an object attribute.

 **Note:**

This signature is **only** available if SDO_GEOmetry (Oracle Locator) is installed in the database.

Syntax

```

APEX_JSON.WRITE (
    p_name      IN VARCHAR2,
    p_value     IN mdsys.sdo_geometry,
    p_write_null IN BOOLEAN DEFAULT FALSE );

```

Parameters

Parameter	Description
p_name	The attribute name.
p_value	The attribute value to be written.
p_write_null	If TRUE, write null values. If FALSE (the default), do not write nulls.

Example

The following example writes a JSON object with the a1, a2, a3, and a4 attributes. a3 is an SDO_GEOmetry, encoded as GeoJSON.

```

DECLARE
    l_sdo_geometry mdsys.sdo_geometry := sdo_geometry( 2001, 4326,
sdo_point_type( 10, 50, null ), null, null );
BEGIN
    apex_json.open_object; -- {
    apex_json.write('a1', 1); -- "a1": 1
    apex_json.write('a2', 'two'); -- , "a2": "two"
    apex_json.write('a3', l_sdo_geometry); -- , "a3": { "type": "Point",
"coordinates": [ 10, 50 ] }
    apex_json.write('a4', false); -- , "a4": false
    apex_json.close_object; -- }
END;

```

37.58 WRITE_CONTEXT Procedure

This procedure writes an array with all rows that the context handle returns. Each row is a separate object.

If the query contains object type, collection or cursor columns, an error is raised. If the column is VARCHAR2 and the uppercase value is 'TRUE' or 'FALSE', boolean values are generated.

Syntax

```
PROCEDURE WRITE_CONTEXT (
    p_name      IN VARCHAR2
    p_context    IN apex_exec.t_context,
    p_write_null IN BOOLEAN  DEFAULT FALSE );
```

Parameters

Parameter	Description
p_name	The attribute name.
p_context	The context handle from an APEX_EXEC.OPEN_QUERY_CONTEXT call.
p_write_null	Whether to write (TRUE) or omit (FALSE) NULL values.

Example

This example opens an APEX_EXEC quey context selecting the DEPT table and passes it to APEX_JSON.

```
DECLARE
    l_context apex_exec.t_context;
begin
    l_context := apex_exec.open_query_context(
        p_location  => apex_exec.c_location_local_db,
        p_sql_query  => q'#select * from dept#' );

    apex_json.open_object;
    apex_json.write_context( p_name => 'departments', p_context => l_context);
    apex_json.close_object;
end;

{ "departments":[
    { "DEPTNO":10 , "DNAME":"ACCOUNTING" , "LOC":"NEW YORK" },
    { "DEPTNO":20 , "DNAME":"RESEARCH" , "LOC":"DALLAS" },
    { "DEPTNO":30 , "DNAME":"SALES" , "LOC":"CHICAGO" },
    { "DEPTNO":40 , "DNAME":"OPERATIONS" , "LOC":"BOSTON" } ] }
```

38

APEX_JWT

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.

Note:

APEX_JWT APIs only support HS256 symmetric encryption algorithm for claim signatures. Asymmetric encryption algorithms such as RS256 are not supported.

- [t_token Record](#)
- [ENCODE Function](#)
- [DECODE Function](#)
- [VALIDATE Procedure](#)

38.1 t_token Record

A `t_token` record contains the decoded parts of a JSON Web Token.

Syntax

```
type t_token is record (
    header      VARCHAR2(32767),
    payload     VARCHAR2(32767),
    signature   VARCHAR2(32767) );
```

Parameters

Parameter	Description
header	The Javascript Object Signing and Encryption (JOSE) header contains cryptographic parameters.
payload	The claims which the token asserts.
signature	The signature of header and payload.

38.2 ENCODE Function

This function encodes and optionally encrypts payload.

Syntax

```
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

Parameter	Description
p_iss	Optional "iss" (Issuer) claim.
p_sub	Optional "sub" (Subject) claim.
p_aud	Optional "aud" (Audience) claim.
p_nbf_ts	Optional "nbf" (Not Before) claim.
p_iat_ts	Optional "iat" (Issued At) claim (default systimestamp).
p_exp_sec	Optional "exp" (Expiration Time) claim, in seconds. The start time is taken from "nbf", "iat" or current time.
p_jti	Optional "jti" (JWT ID) Claim.
p_other_claims	Optional raw JSON with additional claims.
p_signature_key	Optional MAC key for the signature. If not null, a 'HS256' signature is added. This requires Oracle Database 12c or higher. Other signature algorithms are not supported.

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.

```
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;
```

38.3 DECODE Function

This function decodes a raw token value.

Syntax

```
APEX_JWT.DECODE (
    p_value      IN VARCHAR2,
    p_signature_key IN RAW      DEFAULT NULL )
RETURN t_token;
```

Parameters

Parameter	Description
p_value	A raw token value contains 3 base64-encoded parts, which are separated by '.'. The parts are header, payload and signature.
p_signature_key	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.

Returns

A t_token.

Errors Raised

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.eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0
        MjI3Nzk2Mzh9.gzSraSYS8EXBxLN_oWnFSRgCzcmJmMjLiuyu5CSpyHI' );
    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

```

38.4 VALIDATE Procedure

This procedure validates the given token.

Syntax

```
APEX_JWT.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

Parameter	Description
p_token	The JWT.
p_iss	If not null, verify that the "iss" claim equals p_iss.
p_aud	If not null, verify that the single "aud" value equals p_aud. If "aud" is an array, verify that the "azp" (Authorized Party) claim equals p_aud. This is an OpenID extension.
p_leeway_seconds	Fudge factor (in seconds) for comparing "exp" (Expiration Time), "nbf" (Not Before), and "iat" (Issued At) claims.

Raises

VALUE_ERROR: Validation failed, check debug log for details.

Example

Verify that l_value is a valid OpenID ID token.

```

DECLARE
    l_value varchar2(4000) := 'eyJ0 ... NiJ9eyJ1c ... 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;
```

APEX_LANG

You can use APEX_LANG API to translate messages.

- [APPLY_XLIFF_DOCUMENT Procedure](#)
- [CREATE_LANGUAGE_MAPPING Procedure](#)
- [CREATE_MESSAGE Procedure](#)
- [DELETE_LANGUAGE_MAPPING Procedure](#)
- [DELETE_MESSAGE Procedure](#)
- [EMIT_LANGUAGE_SELECTOR_LIST Procedure](#)
- [GET_LANGUAGE_SELECTOR_LIST Function](#)
- [GET_MESSAGE Function](#)
- [GET_XLIFF_DOCUMENT Function](#)
- [LANG Function](#)
- [MESSAGE Function \(Deprecated\)](#)
- [PUBLISH_APPLICATION Procedure](#)
- [SEED_TRANSLATIONS Procedure](#)
- [UPDATE_LANGUAGE_MAPPING Procedure](#)
- [UPDATE_MESSAGE Procedure Signature 1](#)
- [UPDATE_MESSAGE Procedure Signature 2](#)
- [UPDATE_TRANSLATED_STRING Procedure](#)

39.1 APPLY_XLIFF_DOCUMENT Procedure

This procedure applies the specified XLIFF document for the specified language to the translation repository.

Syntax

```
APEX_LANG.APPLY_XLIFF_DOCUMENT (
    p_application_id    IN NUMBER,
    p_language          IN VARCHAR2,
    p_document           IN CLOB )
```

Parameters

Parameter	Description
p_application_id	Application ID of the primary application.
p_language	The IANA language code for the existing translation mapping (such as en-us, fr-ca, ja, he).

Parameter	Description
p_document	The XLIFF document containing the translation.

39.2 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 Oracle APEX 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,
    p_direction_right_to_left   IN BOOLEAN DEFAULT FALSE,
    p_image_directory           IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to create the language mapping. This is the ID of the primary language application.
p_language	The IANA language code for the mapping. Examples include en-us, fr-ca, ja, he.
p_translation_application_id	Unique integer value for the ID of the underlying translated application. This number cannot end in 0.
p_direction_right_to_left	Specify document direction: <ul style="list-style-type: none"> • TRUE - right-to-left • FALSE - left-to-right • NULL - default direction of the language
p_image_directory	Specify the directory where images are stored.

Example

The following example demonstrates the creation of the language mapping for an existing APEX application.

```
BEGIN
  --
  -- If running from SQLcl, we need to set the environment
  -- for the Oracle APEX 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 APEX 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;
/

```

39.3 CREATE_MESSAGE Procedure

Use this procedure to create a translatable text message for the specified application.

Syntax

```
APEX_LANG.CREATE_MESSAGE (
    p_application_id      IN NUMBER,
    p_name                IN VARCHAR2,
    p_language             IN VARCHAR2,
    p_message_text         IN VARCHAR2,
    p_used_in_javascript  IN BOOLEAN DEFAULT FALSE,
    p_comment              IN VARCHAR2 DEFAULT NULL,
    p_metadata             IN CLOB      DEFAULT NULL )
```

Parameters

Parameter	Description
p_application_id	The ID of the application for which you wish to create the translatable text message. This is the ID of the primary language application.
p_name	The name of the translatable text message.
p_language	The IANA language code for the mapping. Examples include en-us, fr-ca, ja, or he.
p_message_text	The text of the translatable text message.

Parameter	Description
p_used_in_javascript	Specify if the message needs to be used directly by JavaScript code (use the apex.lang JavaScript API).
p_comment	Developer comments or notes only visible in the App Builder.
p_metadata	Additional data stored alongside the message. Note: This data is not used by Oracle APEX.

Example

The following example demonstrates the creation of a translatable text message.

```
BEGIN
  --
  -- If running from SQLcl, we need to set the environment
  -- for the APEX 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 APEX
  -- application.
  --
  for c1 in (select workspace_id
              from apex_workspaces
              where workspace = 'HR_DEV') loop
    apex_util.set_security_group_id( c1.workspace_id );
    exit;
  end loop;
  apex_lang.create_message(
    p_application_id => 63969,
    p_name => 'TOTAL_COST',
    p_language => 'en',
    p_message_text => 'The total cost is: %0',
    p_used_in_javascript => true );
  commit;
END;
/
```

39.4 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 Oracle APEX release 4.2.3 and later.

Syntax

```
APEX_LANG.DELETE_LANGUAGE_MAPPING (
    p_application_id      IN NUMBER,
    p_language            IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to delete the language mapping. This is the ID of the primary language application.
p_language	The IANA language code for the existing mapping. Examples include en-us, fr-ca, ja, he.

Example

The following example demonstrates the deletion of the language mapping for an existing APEX application and existing translation mapping.

```
begin
    --
    -- If running from SQLcl, we need to set the environment
    -- for the Oracle APEX 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 APEX 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;
/
```

39.5 DELETE_MESSAGE Procedure

Use this procedure to delete a translatable text message in the specified application.

Syntax

```
APEX_LANG.DELETE_MESSAGE (
    p_id      IN NUMBER )
```

Parameters

Parameter	Description
p_id	The ID of the text message.

Example

The following example demonstrates the deletion of an existing translatable text message.

```
begin
    --
    -- If running from SQLcl, we need to set the environment
    -- for the Oracle APEX 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 APEX
    -- application.
    --
    for c1 in (select workspace_id
                from apex_workspaces
               where workspace = 'HR_DEV') loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;

    -- Locate the ID of the specific message and delete it
    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.delete_message(
            p_id => c1.translation_entry_id );
        commit;
        exit;
    end loop;
end;
/
```

39.6 EMIT_LANGUAGE_SELECTOR_LIST Procedure

This procedure determines which languages the current application is translated into and prints the language selector. You can use this procedure from a PL/SQL region to include a language selector.

Syntax

```
APEX_LANG.EMIT_LANGUAGE_SELECTOR_LIST;
```

Parameters

None.

Example

The following example displays a language selector.

```
BEGIN
    apex_lang.emit_language_selector_list;
END;
```

39.7 GET_LANGUAGE_SELECTOR_LIST Function

This function determines which languages the current application is translated into and returns the language selector as an HTML snippet. You can use this function in a Dynamic Content region to include the language selector.

Syntax

```
APEX_LANG.GET_LANGUAGE_SELECTOR_LIST
    RETURN VARCHAR2;
```

Parameters

None.

Returns

This function returns the language selector as an HTML snippet.

Example

The following example demonstrates how to return the language selector as an HTML snippet.

```
DECLARE
    l_content varchar2;
BEGIN
    l_content := apex_lang.get_language_selector_list;
    RETURN l_content;
END;
```

39.8 GET_MESSAGE Function

Translates text strings (or messages) generated from PL/SQL-stored procedures, functions, triggers, packaged procedures, and functions.

Syntax

```
APEX_LANG.GET_MESSAGE (
    p_name          IN VARCHAR2,
    p_params        IN apex_t_varchar2 DEFAULT apex_t_varchar2(),
    p_lang          IN VARCHAR2      DEFAULT NULL,
```

```
p_application_id      IN NUMBER          DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_name	Name of the message as defined in Text Messages under Shared Components of your application in Oracle APEX.
p_params	List of parameter name value pairs to correspond to %my_name %my_age placeholders in the text message. They are replaced by the value of the named parameter processed left to right.
p_lang	Language code for the message to be retrieved. If not specified, APEX uses the current language for the user as defined in the "Application Language Derived From" attribute.
p_application_id	Specifies the application ID within the current workspace that owns the translated message you wish to return. Useful when coding packages that could be called outside of the scope of APEX such as packages called from a database job.

Example

The following example assumes you have defined a message called GREETING_MSG in your application in English as Good morning %name you are %age and in German as Guten Morgen %name, du bist %age.

The following example demonstrates how to invoke this message from PL/SQL:

```
DECLARE
    l_greetings varchar2( 32767 );
BEGIN
    l_greetings := apex_lang.get_message (
        'GREETING_MSG',
        apex_t_varchar2 (
            'name', :P1_NAME,
            'age',   :P1_AGE )
    );
END;
```

39.9 GET_XLIFF_DOCUMENT Function

This function returns the XLIFF document for the specified language.

Syntax

```
APEX_LANG.GET_XLIFF_DOCUMENT (
    p_application_id      IN NUMBER,
    p_page_id             IN NUMBER  DEFAULT NULL,
    p_language             IN VARCHAR2,
    p_only_modified_elements IN BOOLEAN DEFAULT FALSE )
RETURN CLOB;
```

Parameters

Parameter	Description
p_application_id	Application ID of the primary application.
p_page_id	(Optional) Page ID if the XLIFF document must only contain the specified page.
p_language	The IANA language code for the existing translation mapping (such as en-us, fr-ca, ja, he).
p_only_modified_elements	Choose whether to export all translatable elements of the application or only those elements which are new or have been updated.

39.10 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

Parameter	Description
p_primary_text_string	Text string of the primary language. This is the value of the Translate From Text in the dynamic translation.
p0 through p9	Dynamic substitution value: p0 corresponds to %0 in the translation string; p1 corresponds to %1 in the translation string; p2 corresponds to %2 in the translation string, and so on.
p_primary_language	Language code for the message to be retrieved. If not specified, Oracle APEX 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 <i>Oracle APEX App Builder User's Guide</i> .

Example

In a table that defines all primary colors, you can define a dynamic message for each color and then apply the LANG function to the defined values in a query. For example:

```
SELECT APEX_LANG.LANG(color)
FROM my_colors
```

In an application in German where RED (English) is a value for the color column in the my_colors table, and you defined the German word for red, the previous example returns ROT.

39.11 MESSAGE Function (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use [GET_MESSAGE Function](#) instead.

Use this function to translate text strings (or messages) generated from PL/SQL stored procedures, functions, triggers, packaged procedures, and functions.

Syntax

```
APEX_LANG.MESSAGE (
    p_name          IN VARCHAR2 DEFAULT NULL,
    p0              IN VARCHAR2 DEFAULT NULL,
    p1              IN VARCHAR2 DEFAULT NULL,
    p2              IN VARCHAR2 DEFAULT NULL,
    ...
    p9              IN VARCHAR2 DEFAULT NULL,
    p_lang          IN VARCHAR2 DEFAULT NULL,
    p_application_id IN NUMBER   DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_name	Name of the message as defined in Text Messages under Shared Components of your application in Oracle APEX.
p0 through p9	Dynamic substitution value: p0 corresponds to %0 in the translation string; p1 corresponds to %1 in the translation string; p2 corresponds to %2 in the translation string, and so on.
p_lang	Language code for the message to be retrieved. If not specified, APEX 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 <i>Oracle APEX App Builder User's Guide</i> .
p_application_id	Used to specify the application ID within the current workspace that owns the translated message you wish to return. Useful when coding packages that might be called outside of the scope of APEX such as packages called from a database job.

Example

The following example assumes you have defined a message called GREETING_MSG in your application in English as Good morning %0 and in German as Guten Tag %1. The following example demonstrates how to invoke this message from PL/SQL:

```
BEGIN
--
-- Print the greeting
--
HTP.P(APEX_LANG.MESSAGE('GREETING_MSG', V('APP_USER')));
END;
```

How the p_lang attribute is defined depends on how the APEX 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 APEX 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 APEX engine looks for a message by this name with a language code of en.

 **See Also:**

- [GET_MESSAGE Function](#)
- Specifying the Primary Language for an Application in *Oracle APEX App Builder User's Guide*

39.12 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 Oracle APEX release 4.2.3 and later.

Syntax

```
APEX_LANG.PUBLISH_APPLICATION (
    p_application_id    IN NUMBER,
    p_language          IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	The ID of the application for which you want to publish and create the translated version. This is the ID of the primary language application.
p_language	The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he.

Example

The following example demonstrates the publish process for an APEX application and language.

```

begin
  --
  -- If running from SQLcl, we need to set the environment
  -- for the Oracle APEX 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 APEX application.
  --
  for cl in (select workspace_id
              from apex_workspaces) loop
    apex_util.set_security_group_id( cl.workspace_id );
    exit;
  end loop;
  -- Now, publish the translated version of the application
  apex_lang.publish_application(
    p_application_id => 63969,
    p_language => 'ja' );
  commit;
end;
/

```

39.13 SEED_TRANSLATIONS Procedure

This procedure seeds 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

Parameter	Description
p_application_id	The ID of the application for which you want to update the translation repository. This is the ID of the primary language application.
p_language	The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he.

Example

The following example demonstrates the seeding process of the translation repository for an Oracle APEX application and language.

```

begin
  --
  -- If running from SQLcl, we need to set the environment
  -- for the Oracle APEX 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 APEX application.
  --
  for c1 in (select workspace_id
              from apex_workspaces) loop
    apex_util.set_security_group_id( c1.workspace_id );
    exit;
  end loop;
  -- Now, seed the translation repository
  apex_lang.seed_translations(
    p_application_id => 63969,
    p_language => 'ja' );
  commit;
  -- Print out the total number of potentially translatable strings
  --
  for c1 in (select count(*) thecount
              from apex_application_trans_repos
              where application_id = 63969) loop
    dbms_output.put_line( 'Potentially translatable strings found: ' ||
c1.thecount );
  end loop;
end;
/

```

39.14 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 Oracle APEX 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

Parameters	Description
p_application_id	The ID of the application for which you want to update the language mapping. This is the ID of the primary language application.
p_language	The IANA language code for the existing mapping. Examples include en-us, fr-ca, ja, he. The language of the mapping cannot be updated with this procedure, only the new translation application ID.
p_new_trans_application_id	New unique integer value for the ID of the underlying translated application. This number cannot end in 0.

Example

The following example demonstrates the update of the language mapping for an existing APEX application and existing translation mapping.

```
begin
  --
  -- If running from SQLcl, we need to set the environment
  -- for the Oracle APEX 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 APEX 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;
/

```

39.15 UPDATE_MESSAGE Procedure Signature 1

This procedure updates a translatable text message for the specified application.

An error raises if the message being updated is subscribed.

Note:

When a text message is subscribed, it becomes read-only. In such cases, all changes are driven from the master text message.

Use App Builder to refresh the text message or publish the master text message to get the latest changes from master text message into the text message.

To update the text message using this API, first **unsubscribe** from the text message in App Builder (there is no API for unsubscribing).

Syntax

```
APEX_LANG.UPDATE_MESSAGE (
    p_id          IN NUMBER,
    p_message_text IN VARCHAR2 )
```

Parameters

Parameter	Description
p_id	The ID of the text message.
p_message_text	The new text for the translatable text message.

Example

The following example demonstrates an update of an existing translatable text message.

```
BEGIN
    --
    -- If running from SQLcl, we need to set the environment
    -- for the Oracle APEX 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 APEX 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 = 'en') LOOP
    apex_lang.update_message(
        p_id => c1.translation_entry_id,
        p_message_text => 'The total cost is: %0');
    COMMIT;
    EXIT;
END LOOP;
END;
/

```

See Also:

- Unsubscribing to a Shared Component in the *Oracle APEX App Builder User's Guide*
- Refreshing a Subscribed Shared Component in the *Oracle APEX App Builder User's Guide*

39.16 UPDATE_MESSAGE Procedure Signature 2

This procedure updates a translatable text message and its attributes for the specified application.

Syntax

```
APEX_LANG.UPDATE_MESSAGE (
    p_id          IN NUMBER,
    p_name        IN VARCHAR2,
    p_language    IN VARCHAR2,
    p_message_text IN VARCHAR2,
    p_used_in_javascript IN BOOLEAN,
    p_comment     IN VARCHAR2,
    p_metadata    IN CLOB )
```

Parameters

Parameter	Description
p_id	The ID of the text message to be updated.
p_name	The name of the translatable text message.

Parameter	Description
p_language	The IANA language code for the mapping. Examples include en-us, fr-ca, ja, or he.
p_message_text	The text of the translatable text message.
p_used_in_javascript	Specify if the message needs to be used directly by JavaScript code (use the apex.lang JavaScript API).
p_comment	Developer comments or notes only visible in the App Builder.
p_metadata	Additional data stored alongside with the message. Note: This data is not used by Oracle APEX.

Example

The following example updates a translatable text message.

```
BEGIN
  --
  -- If running from SQLcl, we need to set the environment
  -- for the Oracle APEX workspace associated with this schema.
  -- The call to apex_util.set_workspace is not necessary if
  -- you're running within the context of the App Builder or an APEX
  -- application.
  --
  apex_util.set_workspace( 'SALES_DEV' );

  FOR l_message IN ( select translation_entry_id
                      FROM apex_application_translations
                     WHERE application_id      = 100
                           AND translatable_message = 'TOTAL_COST'
                           AND language_code       = 'en-us' )
  LOOP
    apex_lang.update_message(
      p_id              => l_message.translation_entry_id,
      p_name            => 'SALES_TOTAL_COST',
      p_language        => 'en',
      p_message_text   => 'Total sales cost is: %0',
      p_used_in_javascript => true,
      p_comment         => 'What is the total cost of sales',
      p_metadata        => q'[{"Tag": "sales", "Approved": true}]' );
      -- Any additional data to store
  END LOOP;
END;
```

39.17 UPDATE_TRANSLATED_STRING Procedure

Use this procedure to update a translated string in the seeded translation repository.

 **Note:**

This procedure is available in Oracle APEX 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

Parameter	Description
p_id	The ID of the string in the translation repository.
p_language	The IANA language code for the existing translation mapping. Examples include en-us, fr-ca, ja, he. The language of the mapping cannot be updated with this procedure, only the new translation application ID.
p_string	The new value for the string in the translation repository.

Example

The following example demonstrates an update of an existing string in the translation repository.

```
begin
    --
    -- If running from SQLcl, we need to set the environment
    -- for the Oracle APEX 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 APEX application.
    --
    for c1 in (select workspace_id
                from apex_workspaces) loop
        apex_util.set_security_group_id( c1.workspace_id );
        exit;
    end loop;
    -- Locate all strings in the repository for the specified application
    -- which are 'Search' and change to 'Find'
    for c1 in (select id
                from apex_application_trans_repos
               where application_id = 63969
                 and dbms_lob.compare(from_string, to_nclob('Search')) = 0
                 and language_code = 'en') loop
        apex_lang.update_translated_string(
            p_id => c1.id,
            p_language => 'en',
            p_string => 'Find');
        commit;
        exit;
    end loop;
end;
/
```

40

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](#)

40.1 AUTHENTICATE Function

This 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

Parameter	Description
p_username	Login name of the user.
p_password	Password for p_username.
p_search_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	(Default) Set to N to not use SSL. Set to Y to use SSL in bind to LDAP server.
	Set to A to use SSL with one-way authentication (requires LDAP server certificate configured in an Oracle wallet).

Example

The following example demonstrates how to use the `APEX_LDAP.AUTHENTICATE` function to verify user credentials against an LDAP Server.

```
IF APEX_LDAP.AUTHENTICATE(
    p_username => 'firstname.lastname',
    p_password => 'abcdef',
    p_search_base => 'cn=user,l=amer,dc=example,dc=com',
    p_host => 'our_ldap_sever.example.com',
    p_port => '636',
    p_use_ssl => 'A') THEN

    dbms_output.put_line('authenticated');
ELSE
    dbms_output.put_line('authentication failed');
END IF;
```

40.2 GET_ALL_USER_ATTRIBUTES Procedure

This procedure returns two OUT arrays of `user_attribute` names and values for the user name designated by `p_username` (with password if required) using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.GET_ALL_USER_ATTRIBUTES (
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2 DEFAULT NULL,
    p_host              IN VARCHAR2,
    p_port              IN VARCHAR2 DEFAULT 636,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
    p_attributes        OUT apex_application_global_vc_arr2,
    p_attribute_values  OUT apex_application_global_vc_arr2,
    p_credential_static_id IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
<code>p_username</code>	Login name of the user.
<code>p_pass</code>	Password for <code>p_username</code> .
<code>p_auth_base</code>	LDAP search base, for example, <code>dc=users,dc=my,dc=org</code> .
<code>p_host</code>	LDAP server host name.
<code>p_port</code>	LDAP server port number.
<code>p_use_ssl</code>	(Default) Set to N to not use SSL. Set to Y to use SSL in bind to LDAP server. Set to A to use SSL with one-way authentication (requires LDAP server certificate configured in an Oracle wallet).
<code>p_attributes</code>	An array of attribute names returned.

Parameter	Description
p_attribute_values	An array of values returned for each corresponding attribute name returned in p_attributes.
p_credential_static_id	The credential static ID (can be NULL for anonymous or username/password binds). If it is not NULL and the credential could not be found, then raises the error no_data_found.

Example

The following example demonstrates how to use the APEX_LDAP.GET_ALL_USER_ATTRIBUTES procedure to retrieve all attribute value's associated to a user.

```

DECLARE
    L_ATTRIBUTES apex_application_global.vc_arr2;
    L_ATTRIBUTE_VALUES apex_application_global.vc_arr2;
BEGIN
    APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
        p_username => 'firstname.lastname',
        p_pass => 'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=example,dc=com',
        p_host => 'our_ldap_sever.example.com',
        p_port => '636',
        p_user_ssl => 'A',
        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;

```

40.3 GET_USER_ATTRIBUTES Procedure

This procedure returns an OUT array of user_attribute values for the user name designated by p_username (with password if required) corresponding to the attribute names passed in p_attributes using the provided auth base, host, and port.

Syntax

```

APEX_LDAP.GET_USER_ATTRIBUTES (
    p_username          IN  VARCHAR2 DEFAULT NULL,
    p_pass              IN  VARCHAR2 DEFAULT NULL,
    p_auth_base         IN  VARCHAR2,
    p_host              IN  VARCHAR2,
    p_port              IN  VARCHAR2 DEFAULT 389,
    p_use_ssl           IN  VARCHAR2 DEFAULT 'N',
    p_attributes         IN  apex_application_global.vc_arr2,
    p_attribute_values   OUT apex_application_global.vc_arr2,
    p_credential_static_id IN  VARCHAR2 DEFAULT NULL );

```

Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	(Default) Set to N to not use SSL. Set to Y to use SSL in bind to LDAP server. Set to A to use SSL with one-way authentication (requires LDAP server certificate configured in an Oracle wallet).
p_attributes	An array of attribute names for which values are to be returned.
p_attribute_values	An array of values returned for each corresponding attribute name in p_attributes.
p_credential_static_id	The credential static ID (can be NULL for anonymous or username/password binds). If it is not NULL and the credential could not be found, then raises the error no_data_found.

Example

The following example demonstrates how to use the APEX_LDAP.GET_USER_ATTRIBUTES procedure to retrieve a specific attribute value associated to a user.

```

DECLARE
    L_ATTRIBUTES apex_application_global.vc_arr2;
    L_ATTRIBUTE_VALUES apex_application_global.vc_arr2;
BEGIN
    L_ATTRIBUTES(1) := 'xxxxxxxxxx'; /* name of the employee number attribute
*/
    APEX_LDAP.GET_USER_ATTRIBUTES(
        p_username => 'firstname.lastname',
        p_pass => NULL,
        p_auth_base => 'cn=user,l=amer,dc=example,dc=com',
        p_host => 'our_ldap_sever.example.com',
        p_port => '636',
        p_use_ssl => 'A',
        p_attributes => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);
END;

```

40.4 IS_MEMBER Function

This 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,
    p_credential_static_id IN VARCHAR2 DEFAULT NULL );
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	(Default) Set to N to not use SSL. Set to Y to use SSL in bind to LDAP server. Set to A to use SSL with one-way authentication (requires LDAP server certificate configured in an Oracle wallet).
p_group	Name of the group to be search for membership.
p_group_base	The base from which the search should be started.
p_credential_s tatic_id	The credential static ID (can be NULL for anonymous or username/pass binds). If it is not NULL and the credential could not be found, then raises the error no_data_found.

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=example,dc=com',
        p_host => 'our_ldap_sever.example.com',
        p_port => '636',
        p_use_ssl => 'A',
        p_group => 'group_name',
        p_group_base => 'group_base');
    IF L_VAL THEN
        htp.p('Is a member.');
    END IF;
END;
```

```

        ELSE
            htp.p('Not a member.');
        END IF;
    END;

```

40.5 MEMBER_OF Function

This 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',
    p_credential_static_id IN VARCHAR2 DEFAULT NULL );
RETURN apex_application_global.vc_arr2;

```

Parameters

Parameter	Description
<code>p_username</code>	Login name of the user.
<code>p_pass</code>	Password for <code>p_username</code> .
<code>p_auth_base</code>	LDAP search base, for example, <code>dc=users,dc=my,dc=org</code> .
<code>p_host</code>	LDAP server host name.
<code>p_port</code>	LDAP server port number.
<code>p_use_ssl</code>	(Default) Set to <code>N</code> to not use SSL. Set to <code>Y</code> to use SSL in bind to LDAP server. Set to <code>A</code> to use SSL with one-way authentication (requires LDAP server certificate configured in an Oracle wallet).
<code>p_credential_static_id</code>	The credential static ID (can be <code>NULL</code> for anonymous or username/password binds). If it is not <code>NULL</code> and the credential could not be found, then raises the error <code>no_data_found</code> .

Example

The following example demonstrates how to use the `APEX_LDAP.MEMBER_OF` function to retrieve all the groups designated by the specified username.

```

DECLARE
    L_MEMBERSHIP apex_application_global.vc_arr2;
BEGIN
    L_MEMBERSHIP := APEX_LDAP.MEMBER_OF(
        p_username => 'firstname.lastname',
        p_pass => 'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=example,dc=com',

```

```

    p_host => 'our_ldap_sever.example.com',
    p_port => '636'
    p_use_ssl => 'A');

FOR i IN L_MEMBERSHIP.FIRST..L_MEMBERSHIP.LAST LOOP
    htp.p('Member of: '||L_MEMBERSHIP(i));
END LOOP;
END;

```

40.6 MEMBER_OF2 Function

This 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',
    p_credential_static_id IN VARCHAR2 DEFAULT NULL );
RETURN VARCHAR2;

```

Parameters

Parameter	Description
<code>p_username</code>	Login name of the user.
<code>p_pass</code>	Password for <code>p_username</code> .
<code>p_auth_base</code>	LDAP search base, for example, <code>dc=users,dc=my,dc=org</code> .
<code>p_host</code>	LDAP server host name.
<code>p_port</code>	LDAP server port number.
<code>p_use_ssl</code>	(Default) Set to N to not use SSL. Set to Y to use SSL in bind to LDAP server. Set to A to use SSL with one-way authentication (requires LDAP server certificate configured in an Oracle wallet).
<code>p_credential_static_id</code>	The credential static ID (can be NULL for anonymous or username/pass binds). If it is not NULL and the credential could not be found, then raises the error <code>no_data_found</code> .

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=example,dc=com',
    p_host => 'our_ldap_sever.example.com',
    p_port => '636',
    p_use_ssl => 'A');

http.p('Is Member of:' || L_VAL);
END;

```

40.7 SEARCH Function

The `SEARCH` function searches the LDAP repository and returns an object table of (dn, name, val) that can be used in table queries.

Syntax

```

APEX_LDAP.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,
    p_credential_static_id IN VARCHAR2 DEFAULT NULL )
RETURN apex_t_ldap_attributes pipelined;

```

Parameters

Parameter	Descriptions
p_username	Username to connect as (can be null for anonymous binds).
p_pass	Password of p_username (can be null for anonymous binds).
p_auth_base	Authentication base dn for p_username (can be null for anonymous binds).
p_host	LDAP server hostname.
p_port	LDAP server port (default 389).
p_use_ssl	Y if a SSL connection is required (default N).
p_search_base	dn base for the search.
p_search_filter	LDAP search filter expression.
p_scope	Search scope (default descends into sub-trees).
p_timeout_sec	Timeout for the search (default 3 seconds).
p_attribute_names	Comma-separated list of return attribute names.

Parameter	Descriptions
p_credential_static_id	The credential static ID (can be null for anonymous or username/pass binds). If it is not null and the credential could not be found, then raises the error no_data_found.

Example 1

```
SELECT val group_dns
  FROM table(apex_ldap.search (
    p_host          => 'ldap.example.com',
    p_port          => '636',
    p_use_ssl       => 'A',
    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_port          => '636',
          p_use_ssl       => 'A',
          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 ))
```

APEX_MAIL

You can use the `APEX_MAIL` package to send an email from an Oracle APEX 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 notable 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.

APEX installs the database job `ORACLE_APEX_MAIL_QUEUE`, which periodically sends all mail messages stored in the active mail queue.

Note:

The `APEX_MAIL` package may be used from outside the context of an APEX application (such as from SQLcl or from a Database Scheduler job) as long as the database user making the call is mapped to an APEX workspace. If the database user is mapped to multiple workspaces, you must first call `APEX_UTIL.SET_WORKSPACE` or `APEX_UTIL.SET_SECURITY_GROUP_ID` as in the following examples. The `APEX_MAIL` package cannot be used by database users that are not mapped to any workspace unless they have been granted the role `APEX_ADMINISTRATOR_ROLE`.

```
-- Example 1
apex_util.set_workspace(p_workspace => 'MY_WORKSPACE');

-- Example 2
FOR c1 IN (
    SELECT workspace_id
    FROM apex_applications
    WHERE application_id = 100 )
LOOP
    apex_util.set_security_group_id(p_security_group_id =>
c1.workspace_id);
END LOOP;
```

- [Configuring Oracle APEX to Send Email](#)
- [ADD_ATTACHMENT Procedure Signature 1](#)
- [ADD_ATTACHMENT Procedure Signature 2](#)
- [GET_IMAGES_URL Function](#)
- [GET_INSTANCE_URL Function](#)

- [PREPARE_TEMPLATE Procedure](#)
- [PUSH_QUEUE Procedure](#)
- [SEND Function Signature 1](#)
- [SEND Function Signature 2](#)
- [SEND Procedure Signature 1](#)
- [SEND Procedure Signature 2](#)

 **See Also:**

- [Sending Email from an Application in Oracle APEX App Builder User's Guide](#)
- [Oracle Database PL/SQL Packages and Types Reference](#) for more information about the UTL_SMTP package

41.1 Configuring Oracle APEX to Send Email

Before you can send email from an App Builder application, you must:

1. Log in to APEX Administration Services and configure the email settings on the Instance Settings page. See [Configuring Email in Oracle APEX Administration Guide](#).
2. Enable network services that are disabled by default in Oracle Database 11g release 2 (11.2) and newer. See [Enabling Network Service in Oracle Database 11g](#) in [Enabling Network Services in Oracle Database 11g or Later](#) in [Oracle APEX App Builder User's Guide](#).

 **Tip:**

You can configure APEX 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 APEX Administration Guide](#).

41.2 ADD_ATTACHMENT Procedure Signature 1

This procedure adds an attachment of type BLOB to an outbound email message. 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
    p_content_id       IN VARCHAR2    DEFAULT NULL );
```

Parameters

Parameter	Description
p_mail_id	The numeric ID associated with the email. This is the numeric identifier returned from the call to APEX_MAIL.SEND to compose the email body.
p_attachment	A BLOB variable containing the binary content to be attached to the email message.
p_filename	The filename associated with the email attachment.
p_mime_type	A valid MIME type (or Internet media type) to associate with the email attachment.
p_content_id	An optional identifier for the attachment. If non-null, then the file attaches inline. That attachment may then be referenced in the HTML of the email body by using the cid. Note: Be aware that automatic displaying of inlined images may not be supported by all e-mail clients.

Example 1

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;
/

```

Example 2

This example shows how to attach a file inline, by using a content identifier, and how to refer to that attachment in the HTML of the email.

```

DECLARE
    l_id number;
    l_body clob;

```

```
l_body_html clob;
l_content_id varchar2(100) := 'my-inline-image';
l_filename varchar2(100);
l_mime_type varchar2(100);
l_image blob;
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>Here is the image you requested.</p>' || utl_tcp.crlf ||
                    '<p></p>' || utl_tcp.crlf ||
                    '<p>Thanks,<br />' || utl_tcp.crlf ||
                    'The EveryCorp Dev Team<br />' || utl_tcp.crlf ||
                    '</body></html>';

    l_id := apex_mail.send (
        p_to => 'some_user@example.com', -- change to your email address
        p_from => 'some_sender@example.com', -- change to a real senders email
        address
        p_body => l_body,
        p_body_html => l_body_html,
        p_subj => 'Requested Image' );

    select filename, mime_type, blob_content
    into l_filename, l_mime_type, l_image
    from apex_application_files
    where id = 123;

    apex_mail.add_attachment(
        p_mail_id => l_id,
        p_attachment => l_image,
        p_filename => l_filename,
        p_mime_type => l_mime_type,
        p_content_id => l_content_id );

    COMMIT;
END;
```

41.3 ADD_ATTACHMENT Procedure Signature 2

This procedure adds an attachment of type CLOB to an outbound email message. 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 CLOB,
    p_filename         IN VARCHAR2,
    p_mime_type        IN VARCHAR2 );
```

Parameters

Parameter	Description
p_mail_id	The numeric ID associated with the email. This is the numeric identifier returned from the call to APEX_MAIL.SEND to compose the email body.
p_attachment	A CLOB variable containing the text content to be attached to the email message.
p_filename	The filename associated with the email attachment.
p_mime_type	A valid MIME type (or Internet media type) to associate with the email attachment.

Examples

The following example demonstrates how to attached a CLOB-based attachment to an outbound email message.

```

DECLARE
    l_id NUMBER;
    l_clob CLOB := 'Value1,Value2,Value3,42';
BEGIN
    l_id := APEX_MAIL.SEND(
        p_to => 'fred@flintstone.com',
        p_from => 'barney@rubble.com',
        p_subj => 'APEX_MAIL with a text attachment',
        p_body => 'Please review the attachment.',
        p_body_html => '<b>Please</b> review the attachment');

    APEX_MAIL.ADD_ATTACHMENT(
        p_mail_id => l_id,
        p_attachment => l_clob,
        p_filename => 'data.csv',
        p_mime_type => 'text/csv');

    COMMIT;
END;
/

```

41.4 GET_IMAGES_URL Function

This function gets the image prefixed URL if the email includes Oracle APEX instance images.

Syntax

```
APEX_MAIL.GET_IMAGES_URL return VARCHAR2;
```

Parameters

None.

Example

The following example sends an Order Confirmation email which includes the Oracle Logo image.

```

DECLARE
    l_body      clob;
    l_body_html clob;
BEGIN
    l_body := 'To view the content of this message, please use an HTML
enabled mail client.' || utl_tcp.crlf;

    l_body_html := '<html><body>' || utl_tcp.crlf ||
                   '<p>Please confirm your order on the <a href="' || 
                   apex_mail.get_instance_url || 'f?p=100:10">Order
Confirmation</a> page.</p>' || utl_tcp.crlf ||
                   '<p>Sincerely,<br />' || utl_tcp.crlf ||
                   'The EveryCorp Dev Team<br />' || utl_tcp.crlf ||
                   '</p>' || utl_tcp.crlf ||
                   '</body></html>';
    apex_mail.send (
        p_to      => 'some_user@example.com', -- change to your email
address
        p_from    => 'some_sender@example.com', -- change to a real senders
email address
        p_body    => l_body,
        p_body_html => l_body_html,
        p_subj   => 'Order Confirmation' );
END;

```

41.5 GET_INSTANCE_URL Function

This function gets the instance URL if an email includes a link to an Oracle APEX instance.

 **Note:**

This function requires that the APEX Instance URL parameter is set on the Manage Instance, Instance Settings page in the Email section in Administration Services.

Syntax

```
APEX_MAIL.GET_INSTANCE_URL return VARCHAR2;
```

Parameters

None.

Example

The following example sends an Order Confirmation email which includes an absolute URL to page 10 of application 100.

```

DECLARE
    l_body      clob;
    l_body_html clob;
BEGIN
    l_body := 'To view the content of this message, please use an HTML
enabled mail client.' || utl_tcp.crlf;

    l_body_html := '<html><body>' || utl_tcp.crlf ||
                   '<p>Please confirm your order on the <a href="' || 
                   apex_mail.get_instance_url || 'f?p=100:10">Order
Confirmation</a> page.</p>' || utl_tcp.crlf ||
                   '</body></html>';
    apex_mail.send (
        p_to          => 'some_user@example.com', -- change to your email
address
        p_from         => 'some_sender@example.com', -- change to a real senders
email address
        p_body         => l_body,
        p_body_html   => l_body_html,
        p_subj        => 'Order Confirmation' );
END;

```

 **See Also:**

- Configuring Email in *Oracle APEX Administration Guide*

41.6 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

```
APEX_MAIL.PREPARE_TEMPLATE (
    p_static_id      IN  VARCHAR2,
    p_placeholders   IN  CLOB,
    p_application_id IN  NUMBER  DEFAULT apex_application.g_flow_id,
    p_subject        OUT VARCHAR2,
    p_html           OUT CLOB,
    p_text            OUT CLOB,
    p_language_override IN  VARCHAR2 DEFAULT NULL );
```

Parameters

Parameters	Description
p_static_id	The identifier which was specified when the template was created in the Oracle APEX Builder.
p_placeholders	A JSON formatted string containing name/value pairs specifying values for the placeholders to be replaced in the email template.
p_application_id	Application ID where the email template is defined. Defaults to the current application (if called from within an application).
p_subject	The subject line generated from the template, after any placeholders and substitutions have been made.
p_html	The HTML code for the email, after placeholders have been replaced.
p_text	The plain text of the email, with substitutions made.
p_language_override	Language of a translated template to use. Use a language code like "en", "fr" or "de-at" here. An application translation for this language must exist, otherwise the argument is ignored.

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;
```

41.7 PUSH_QUEUE Procedure

This procedure manually delivers queued mail messages stored in the `APEX_MAIL_QUEUE` dictionary view to the SMTP gateway.

Oracle APEX logs successfully submitted messages in the `APEX_MAIL_LOG` dictionary view 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

Parameters	Description
p_smtp_hostname	SMTP gateway host name
p_smtp_portno	SMTP gateway port number

 **Note:**

Note 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 Instance Settings page in Administration Services or set using `APEX_INSTANCE_ADMIN` API.

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 / <<EOF
APEX_MAIL.PUSH_QUEUE;
DISCONNECT
EXIT
EOF
```

 **See Also:**

- Configuring Email in *Oracle APEX Administration Guide*
- Sending an Email from an Application in *Oracle APEX App Builder User's Guide*

41.8 SEND Function Signature 1

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.

Usage Notes

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 `` 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:

```

```

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.

Alternatively, you may specify the `p_content_id` parameter when calling `APEX_MAIL.ADD_ATTACHMENT` which creates an inline attachment that can be referenced as follows:

```

```

Note that this may greatly increase the size of the resultant emails and that clients may not always automatically display inline images.

For these reasons, 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 nor displayed.

Syntax

```
APEX_MAIL.SEND (
    p_to          IN      VARCHAR2,
    p_from        IN      VARCHAR2,
    p_body        IN      [ VARCHAR2 | CLOB ],
    p_body_html   IN      [ VARCHAR2 | CLOB ] DEFAULT NULL,
    p_subj       IN      VARCHAR2 DEFAULT NULL,
    p_cc          IN      VARCHAR2 DEFAULT NULL,
    p_bcc         IN      VARCHAR2 DEFAULT NULL,
    p_replyto    IN      VARCHAR2 DEFAULT NULL )
RETURN NUMBER;
```

Parameters

Parameter	Description
<code>p_to</code>	Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list

Parameter	Description
p_from	Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent. If an instance administrator has already defined a "Default Email From Address" on the instance level, this address is always used as the "from" address.
p_body	Body of the email in plain text, not HTML (required). If a value is passed to p_body_html, then this is the only text the recipient sees. If a value is not passed to p_body_html, then this text only displays for email clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters.
p_body_html	Body of the email in HTML format. This must be a full HTML document including the <html> and <body> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)
p_subj	Subject of the email
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list
p_replyto	Specify a valid email address to instruct recipient's email client to send human-generated replies to this address rather than the address specified in p_from.

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application and return the unique message ID.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
    l_id NUMBER;
BEGIN
    l_body := 'Thank you for your interest in the APEX_MAIL
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||' Sincerely,'||utl_tcp.crlf;
    l_body := l_body ||' The EveryCorp Dev Team'||utl_tcp.crlf;
    l_id   := apex_mail.send(
        p_to      => 'some_user@example.com', -- change to your email
address
        p_from    => 'some_sender@example.com', -- change to a real senders
email address
        p_body    => l_body,
        p_subj   => 'APEX_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use APEX_MAIL.SEND to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses utl_tcp.crlf.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
    l_id NUMBER;
BEGIN
    l_body := 'To view the content of this message, please use an HTML
enabled mail client.'||utl_tcp.crlf;

    l_body_html := '<html>
        <head>
            <style type="text/css">
                body{font-family: Arial, Helvetica, sans-serif;
                    font-size:10pt;
                    margin:30px;
                    background-color:#ffffff;}
                span.sig{font-style:italic;
                    font-weight:bold;
                    color:#800000;}
            </style>
        </head>
        <body>'||utl_tcp.crlf;
    l_body_html := l_body_html || '<p>Thank you for your interest in the
<strong>APEX_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html || ' Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html || '<span class="sig">The EveryCorp Dev Team</
span><br />'||utl_tcp.crlf;
    l_body_html := l_body_html || '</body></html>';
    l_id      := apex_mail.send(
        p_to      => 'some_user@example.com', -- change to your email
address
        p_from    => 'some_sender@example.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;
/

```

41.9 SEND Function Signature 2

This function returns a mail ID after adding the mail to the mail queue of APEX. 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 email template where the placeholder values specified as JSON string are substituted.

Syntax

```
APEX_MAIL.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 apex_application.g_flow_id,
    p_language_override        IN VARCHAR2 DEFAULT NULL );
    RETURN NUMBER;
```

Parameters

Parameter	Description
p_template_static_id	Static identifier string, used to identify the shared component email template.
p_placeholders	JSON string representing the placeholder names along with the values, to be substituted.
p_to	Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list.
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list.
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list.
p_from	This must be a valid email address from which the email is sent. If p_from is not provided: <ul style="list-style-type: none"> • the "Application Email From Address" is used (if provided under Application Definition) • otherwise, p_replyto is used (if provided) • else, p_to is used If p_to or p_replyto contain multiple email addresses, p_from is required. If an instance administrator has already defined a "Default Email From Address" on the instance level, this address is always used as the "from" address, regardless of whether "p_from" is provided.

Parameter	Description
p_replyto	<p>Specify a valid email address to instruct recipient's email client to send human-generated replies to this address rather than the address specified in p_from. You can use this parameter as follows:</p> <ul style="list-style-type: none"> 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).
p_application_id	Application ID where the email template is defined. Defaults to the current application (if called from within an application).
p_language_override	Language of a translated template to use. Use a language code like "en", "fr" or "de-at" here. An application translation for this language must exist, otherwise the argument is ignored.

Note:

When calling the SEND function from outside the context of an APEX application (such as from a Database Scheduler job), you must specify the p_application_id parameter.

Examples

```

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@example.com' );

    apex_mail.add_attachment (
        p_mail_id    => l_mail_id,
        p_attachment => ... );
END;

```

41.10 SEND Procedure Signature 1

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`.

Usage Notes

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 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 `` 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:

```

```

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.

Alternatively, you may specify the `p_content_id` parameter when calling `APEX_MAIL.ADD_ATTACHMENT` which creates an inline attachment that can be referenced as follows:

```

```

Note that this may greatly increase the size of the resultant emails and that clients may not always automatically display inline images.

For these reasons, 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 nor displayed.

Syntax

```
APEX_MAIL.SEND (
    p_to          IN      VARCHAR2,
    p_from        IN      VARCHAR2,
    p_body        IN      [ VARCHAR2 | CLOB ],
    p_body_html   IN      [ VARCHAR2 | CLOB ] DEFAULT NULL,
    p_subj        IN      VARCHAR2 DEFAULT NULL,
    p_cc          IN      VARCHAR2 DEFAULT NULL,
```

```
p_bcc          IN      VARCHAR2 DEFAULT NULL,
p_replyto     IN      VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_to	Valid email address to which the email is sent (required). For multiple email addresses, use a comma-separated list.
p_from	Email address from which the email is sent (required). This email address must be a valid address. Otherwise, the message is not sent.
p_body	If an instance administrator has already defined a "Default Email From Address" on the instance level, this address is always used as the "from" address.
p_body_html	Body of the email in plain text, not HTML (required). If a value is passed to p_body_html, then this is the only text the recipient sees. If a value is not passed to p_body_html, then this text only displays for email clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters.
p_subj	Body of the email in HTML format. This must be a full HTML document including the <html> and <body> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)
p_cc	Subject of the email.
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list.
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list.
p_replyto	Specify a valid email address to instruct recipient's email client to send human-generated replies to this address rather than the address specified in p_from.

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
BEGIN
    l_body := 'Thank you for your interest in the APEX_MAIL
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||' Sincerely,'||utl_tcp.crlf;
    l_body := l_body ||' The EveryCorp Dev Team'||utl_tcp.crlf;
    apex_mail.send(
        p_to      => 'some_user@example.com', -- change to your email
address
        p_from    => 'some_sender@example.com', -- change to a real senders
email address
        p_body    => l_body,
        p_subj   => 'APEX_MAIL Package - Plain Text message');
```

```
END;
/
```

The following example demonstrates how to use APEX_MAIL.SEND to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses utl_tcp.crlf.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
BEGIN
    l_body := 'To view the content of this message, please use an HTML
enabled mail client.'||utl_tcp.crlf;

    l_body_html := '<html>
        <head>
            <style type="text/css">
                body{font-family: Arial, Helvetica, sans-serif;
                    font-size:10pt;
                    margin:30px;
                    background-color:#ffffff;}
                span.sig{font-style:italic;
                    font-weight:bold;
                    color:#811919;}
            </style>
        </head>
        <body>'||utl_tcp.crlf;
    l_body_html := l_body_html || '<p>Thank you for your interest in the
<strong>APEX_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html || ' Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html || '<span class="sig">The EveryCorp Dev Team</
span><br />'||utl_tcp.crlf;
    l_body_html := l_body_html || '</body></html>';
    apex_mail.send(
        p_to  => 'some_user@example.com', -- change to your email address
        p_from => 'some_sender@example.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;
/
```

41.11 SEND Procedure Signature 2

This procedure adds a mail to the mail queue of Oracle APEX. The mail is based on an email template where the placeholder values specified as JSON string are substituted.

Syntax

```
APEX_MAIL.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 apex_application.g_flow_id );

```

Parameters

Parameter	Description
p_template_static_id	Static identifier string, used to identify the shared component email template.
p_placeholders	JSON string representing the placeholder names along with the values, to be substituted.
p_to	(Required) Valid email address to which the email is sent. For multiple email addresses, use a comma-separated list.
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list.
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list.
p_from	This must be a valid email address from which the email is sent. If p_from is not provided: <ul style="list-style-type: none"> • the "Application Email From Address" is used (if provided under Application Definition) • otherwise, p_replyto is used (if provided) • else, p_to is used If p_to or p_replyto contain multiple email addresses, p_from is required. If an instance administrator has already defined a "Default Email From Address" on the instance level, this address is always used as the "from" address, regardless of whether p_from is provided.
p_replyto	Specify a valid email address to instruct recipient's email client to send human-generated replies to this address rather than the address specified in p_from. You can use this parameter as follows: <ul style="list-style-type: none"> • 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 disables 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)
p_application_id	Application ID where the email template is defined. Defaults to the current application (if called from within an application).

Note:

When calling the SEND procedure from outside the context of an APEX application (such as from a Database Scheduler job), you must specify the p_application_id parameter.

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@example.com' );
end;
```

APEX_MARKDOWN

This package offers a way to convert Markdown to HTML directly in the database.

This parser is compliant with the [CommonMark Spec version 0.29](#).

- [Constants](#)
- [TO_HTML Function](#)

42.1 Constants

The following constants are used by this package.

```
c_embedded_html_escape    constant t_embedded_html_mode := 'ESCAPE';
-- escapes HTML
c_embedded_html_preserve constant t_embedded_html_mode := 'PRESERVE';
-- leaves HTML content as-is
```

42.2 TO_HTML Function

This function converts a Markdown string into HTML.

Syntax

```
APEX_MARKDOWN.TO_HTML (
    p_markdown          IN CLOB,
    p_embedded_html_mode IN t_embedded_html_mode DEFAULT
c_embedded_html_escape,
    p_softbreak         IN VARCHAR2                  DEFAULT '<br />',
    p_extra_link_attributes IN apex_t_varchar2      DEFAULT
apex_t_varchar2() )
    RETURN CLOB;
```

Parameters

Parameter	Description
p_markdown	The Markdown text content to be converted to HTML.
p_embedded_html_mode	Specify what should happen with embedded HTML. By default it is escaped.
p_softbreak	Set this option to C_EMBEDDED_HTML_PRESERVE for it to be preserved. Note that this option has security implications and should only ever be used on trusted input.
p_extra_link_attributes	Specify a raw string to be used for a softbreak, such as apex_application.LF. If none is specified, uses .

Parameter	Description
p_extra_link_attributes	A plist of additional HTML attributes for anchor elements. For example, to open all links in new tabs, set this parameter to apex_t_varchar2('target', '_blank')

Example

```
DECLARE
    l_markdown varchar2(100) := '## APEX_MARKDOWN' || chr(10) || '- Includes
the `to_html` **function**';
BEGIN
    dbms_output.put_line(apex_markdown.to_html(l_markdown));
END;
```

43

APEX_PAGE

The APEX_PAGE package is the public API for handling pages.

- [Global Constants](#)
- [GET_PAGE_MODE Function](#)
- [GET_UI_TYPE Function \(Deprecated\)](#)
- [GET_URL Function](#)
- [ISSKTOP_UI Function \(Deprecated\)](#)
- [IS_READ_ONLY Function](#)
- [PURGE_CACHE Procedure](#)

43.1 Global Constants

The APEX_PAGE package uses the following constants.

```
c_ui_type_desktop    constant varchar2(10) := 'DESKTOP'; -- Deprecated
```

43.2 GET_PAGE_MODE Function

This function returns the page mode for a given page.

Syntax

```
FUNCTION GET_PAGE_MODE (
    p_application_id IN NUMBER,
    p_page_id         IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application_id	ID of the application.
p_page_id	ID of the page.

43.3 GET_UI_TYPE Function (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

This function returns the user interface (UI) type for which the current page has been designed.

Syntax

```
FUNCTION GET_UI_TYPE
RETURN VARCHAR2;
```

43.4 GET_URL Function

Returns an APEX f?p= URL. It is sometimes clearer to read a function call than a concatenated URL. See the example below for a comparison.

Note:

The URL excludes a checksum if the specific application is located in a different workspace.

Syntax

```
APEX_PAGE.GET_URL (
    p_application      IN VARCHAR2 DEFAULT NULL,
    p_page             IN VARCHAR2 DEFAULT NULL,
    p_session          IN NUMBER   DEFAULT apex_application.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,
    p_x01              IN VARCHAR2 DEFAULT NULL,
    p_hash              IN VARCHAR2 DEFAULT NULL,
    p_triggering_element IN VARCHAR2 DEFAULT 'this',
    p_plain_url        IN BOOLEAN  DEFAULT FALSE,
    p_absolute_url     IN BOOLEAN  DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_application	The application ID or alias. Defaults to the current application.
p_page	Page ID or alias. Defaults to the current page.
p_session	Session ID. Defaults to the current session ID.
p_request	URL request parameter.
p_debug	URL debug parameter. Defaults to the current debug mode.
p_clear_cache	URL clear cache parameter.
p_items	Comma-delimited list of item names to set session state.
p_values	Comma-separated list of item values to set session state.

Parameter	Description
p_printer_friendly	URL printer-friendly parameter. Defaults to the current request's printer-friendly mode.
p_trace	SQL trace parameter.
p_x01	Adds the parameter &x01=value to the URL.
p_hash	Adds #hash-value at the end of the URL.
p_triggering_element	A jQuery selector to identify which element to use to trigger the dialog (for example, #my_button, where "my_button" is the static ID for a button element). Required for Modal Dialog support.
p_plain_url	If the page you are calling APEX_PAGE.GET_URL from is a modal dialog, specify p_plain_url to omit the unnecessary JavaScript code in the generated link. By default, if this function is called from a modal dialog, JavaScript code to close the modal dialog is included in the generated URL.
p_absolute_url	If FALSE (default), auto-determines if an absolute URL is needed. If TRUE, always generates an absolute URL.

Example

This query uses APEX_PAGE.GET_URL and its alternative APEX_UTIL.PREPARE_URL to produce two identical URLs.

```
SELECT APEX_PAGE.GET_URL (
    p_page    => 1,
    p_items   => 'P1_X,P1_Y',
    p_values  => 'somevalue,othervalue' ) f_url_1,
APEX_UTIL.PREPARE_URL('f?
p=&APP_ID.:1:&APP_SESSION.:::::P1_X,P1_Y:somevalue,othervalue') f_url_2
FROM DUAL;
```

43.5 IS_DESKTOP_UI Function (Deprecated)



Note:

This API is deprecated and will be removed in a future release.

This function returns TRUE if the current page has been designed for desktop browsers.

Syntax

```
FUNCTION IS_DESKTOP_UI
RETURN BOOLEAN;
```

43.6 IS_READONLY 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;
```

43.7 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

```
APEX_PAGE.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

Parameter	Description
p_application_id	ID of the application. Defaults to the current application.
p_page_id	ID of the page. Defaults to the current page. If you pass NULL, Oracle APEX purges the cache on all pages of the application.
p_user_name	Specify a user name if you only want to purge entries that were saved for the given user.
p_current_session_only	Specify TRUE if you only want to purge entries that were saved for the current session. Defaults to FALSE.

Example

This example purges session specific cache on the current page.

```
BEGIN
    APEX_PAGE.PURGE_CACHE (
        p_current_session_only => true );
END;
```

APEX_PLUGIN

The `APEX_PLUGIN` package provides the interface declarations and some utility functions to work with plug-ins.

- [About Configuring Flexible Remote Servers in APEX](#)
Flexible remote servers use a configuration procedure to change the URL endpoint of the server.
- [Constants](#)
- [Data Types](#)
- [GET_AJAX_IDENTIFIER Function](#)
- [GET_INPUT_NAME_FOR_PAGE_ITEM Function \(Deprecated\)](#)

44.1 About Configuring Flexible Remote Servers in APEX

Flexible remote servers use a configuration procedure to change the URL endpoint of the server.

The procedure is defined by the end user either as a package procedure in the database or as the attribute `PL/SQL Code` of the Remote Server.

The name of this procedure is specified by the end user while editing the remote server. This procedure needs to have a specific signature using `t_remote_server_info` and `t_remote_server_config`.

A resulting procedure resembles the following code example:

```
procedure my_server_config(
    p_info    in apex_plugin.t_remote_server_info,
    p_config  out apex_plugin.t_remote_server_config )
is
begin
    if p_info.application_id = 100
    then
        p_config.base_url := 'https://#cust#.example.com';
        p_config.substitutions := apex_t_varchar2();
        apex_string.plist_put( p_config.substitutions, 'cust',
v('P3_CUSTOMER') );
    else
        p_config.base_url := 'https://test.example.com';
    end if;
end;
```

44.2 Constants

Data Format Constants

The following data format constants are used with REST Data Sources in APEX_PLUGIN:

```
subtype t_data_format           is pls_integer range 1..2;  
  
c_format_xml                  constant t_data_format := 1;  
c_format_json                 constant t_data_format := 2;
```

Database Operation Constants

The following constants are used with REST Data Sources in APEX_PLUGIN:

```
subtype t_db_operation          is pls_integer range 1..6;  
  
c_db_operation_fetch_rows     constant t_db_operation      := 1;  
c_db_operation_insert         constant t_db_operation      := 2;  
c_db_operation_update         constant t_db_operation      := 3;  
c_db_operation_delete        constant t_db_operation      := 4;  
c_db_operation_fetch_row     constant t_db_operation      := 5;  
c_db_operation_execute       constant t_db_operation      := 6;
```

REST Data Source Parameter Constants

The following constants are used with REST Data Sources in APEX_PLUGIN:

```
subtype t_web_source_param_type is pls_integer range 1..5;  
  
c_web_src_param_header       constant t_web_source_param_type := 1;  
c_web_src_param_query        constant t_web_source_param_type := 2;  
c_web_src_param_url_pattern constant t_web_source_param_type := 3;  
c_web_src_param_body         constant t_web_source_param_type := 4;  
c_web_src_param_cookie       constant t_web_source_param_type := 5;  
  
subtype t_web_source_param_dir is pls_integer range 1..3;  
  
c_direction_in               constant t_web_source_param_dir  := 1;  
c_direction_out              constant t_web_source_param_dir  := 2;  
c_direction_in_out           constant t_web_source_param_dir  := 3;
```

REST Data Source DML Row Status Constants

The following constants are used with REST Data Sources in APEX_PLUGIN:

```
subtype t_web_source_row_check_result is pls_integer range 1..5;  
  
c_row_ok                     constant t_web_source_row_check_result := 1;  
c_row_version_changed         constant t_web_source_row_check_result := 2;  
c_row_data_not_changed        constant t_web_source_row_check_result := 3;  
c_row_refetch_error           constant t_web_source_row_check_result := 4;  
c_row_dml_not_allowed        constant t_web_source_row_check_result := 5;
```

44.3 Data Types

This section describes the data types used by the `APEX_PLUGIN` package.

- `c_inline_in_notification`
- `c_inline_with_field`
- `c_inline_with_field_and_notif`
- `c_on_error_page`
- `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_escape_mode`
- `t_item`
- `t_item_ajax_result`
- `t_item_meta_data_result`
- `t_item_render_param`
- `t_item_render_result`
- `t_item_validation_result`
- `t_plugin`
- `t_plugin_attributes`
- `t_process`
- `t_process_exec_result`
- `t_region`
- `t_region_ajax_result`
- `t_region_column`
- `t_region_columns`
- `t_region_render_param`
- `t_region_render_result`
- `t_remote_server_config`
- `t_remote_server_info`

44.3.1 c_inline_in_notification

Use the following constant for `display_location` in the page item validation function result type `t_page_item_validation_result`.

```
c_inline_in_notification      constant varchar2(40) :=  
'INLINE_IN_NOTIFICATION';
```

44.3.2 c_inline_with_field

Use the constant `c_inline_with_field` for `display_location` in the page item validation function result type `t_page_item_validation_result`.

```
c_inline_with_field      constant varchar2(40) := 'INLINE_WITH_FIELD';
```

44.3.3 c_inline_with_field_and_notif

Use the constant `c_inline_with_field_and_notif` for `display_location` in the page item validation function result type `t_page_item_validation_result`.

```
c_inline_with_field_and_notif  constant varchar2(40) :=  
'INLINE_WITH_FIELD_AND_NOTIFICATION';
```

44.3.4 c_on_error_page

Use the constant `c_on_error_page` for `display_location` in the page item validation function result type `t_page_item_validation_result`.

```
c_on_error_page      constant varchar2(40) := 'ON_ERROR_PAGE';
```

44.3.5 t_authentication

```
type t_authentication is record (  
    id          number,  
    name        varchar2(255),  
    invalid_session_url  varchar2(4000),  
    logout_url   varchar2(4000),  
    plsql_code   clob,  
    attribute_01  varchar2(32767),  
    attribute_02  varchar2(32767),  
    attribute_03  varchar2(32767),  
    attribute_04  varchar2(32767),  
    attribute_05  varchar2(32767),  
    attribute_06  varchar2(32767),  
    attribute_07  varchar2(32767),
```

```
attribute_08      varchar2(32767),
attribute_09      varchar2(32767),
attribute_10      varchar2(32767),
attribute_11      varchar2(32767),
attribute_12      varchar2(32767),
attribute_13      varchar2(32767),
attribute_14      varchar2(32767),
attribute_15      varchar2(32767),
--
session_id        number,
username          varchar2(255) );
```

44.3.6 t_authentication_ajax_result

```
type t_authentication_ajax_result is record (
    dummy           boolean );
```

44.3.7 t_authentication_auth_result

```
type t_authentication_auth_result is record (
    is_authenticated   boolean,
    redirect_url       varchar2(4000),
    log_code           number,
    log_text           varchar2(4000),
    display_text       varchar2(4000) );
```

44.3.8 t_authentication_inval_result

```
type t_authentication_inval_result is record (
    redirect_url      varchar2(4000) );
```

44.3.9 t_authentication_logout_result

```
type t_authentication_logout_result is record (
    redirect_url      varchar2(4000) );
```

44.3.10 t_authentication_sentry_result

```
type t_authentication_sentry_result is record (
    is_valid           boolean );
```

44.3.11 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),
```

44.3.12 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
);
```

44.3.13 t_dynamic_action

The `t_dynamic_action` 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,
    name          varchar2(255),
    action        varchar2(50),
    attribute_01  varchar2(32767),
    attribute_02  varchar2(32767),
    attribute_03  varchar2(32767),
    attribute_04  varchar2(32767),
    attribute_05  varchar2(32767),
    attribute_06  varchar2(32767),
    attribute_07  varchar2(32767),
    attribute_08  varchar2(32767),
    attribute_09  varchar2(32767),
    attribute_10  varchar2(32767),
    attribute_11  varchar2(32767),
    attribute_12  varchar2(32767),
    attribute_13  varchar2(32767),
    attribute_14  varchar2(32767),
    attribute_15  varchar2(32767),
```

```
init_javascript_code varchar2(32767),  
triggering_region_id number,  
affected_elements_type varchar2(30),  
affected_region_id number,  
affected_button_id number,  
affected_elements varchar2(4000) );
```

44.3.14 t_dynamic_action_ajax_result

The `t_dynamic_action_ajax_result` type is used as the result type for the Ajax function of a dynamic action type plug-in.

```
type t_dynamic_action_ajax_result is record (  
    dummy boolean /* not used yet */  
) ;
```

44.3.15 t_dynamic_action_render_result

The `t_dynamic_action_render_result` 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) );
```

44.3.16 t_escape_mode

Modes for escaping substitution variables.

```
subtype t_escape_mode is pls_integer range 1..6;  
  
c_escape_mode_raw           constant t_escape_mode := 1;  
c_escape_mode_html          constant t_escape_mode := 2;  
c_escape_mode_html_attribute constant t_escape_mode := 3;  
c_escape_mode_javascript    constant t_escape_mode := 4;  
c_escape_mode_stripthtml   constant t_escape_mode := 5;  
c_escape_mode_json          constant t_escape_mode := 6;
```

44.3.17 t_item

The `t_item` type is passed into all item type plug-in functions and contains information about the current page item.

```
type t_item is record (
    id                               NUMBER,
    name                             VARCHAR2(4000),
    session_state_name              VARCHAR2(4000),
    component_type_id               NUMBER,
    region_id                        NUMBER,
    form_region_id                  NUMBER,
    data_type                         VARCHAR2(32767),
    source_data_type                 apex_exec_api.t_data_type,
    session_state_data_type          apex_session_state_api.t_data_type,
    multi_value_type                 apex_exec_api.t_multi_value_type,
    multi_value_separator            VARCHAR2(10),
    label                            VARCHAR2(4000),
    plain_label                      VARCHAR2(4000),
    label_id                          VARCHAR2(4000),
    placeholder                      VARCHAR2(4000),
    format_mask                      VARCHAR2(4000),
    is_required                      BOOLEAN,
    lov_type                          VARCHAR2(4000),
    lov_definition                   VARCHAR2(4000),
    lov_language                     apex_code_exec.t_language,
    shared_lov_id                    NUMBER,
    lov_display_extra                BOOLEAN,
    lov_display_null                 BOOLEAN,
    lov_null_text                    VARCHAR2(4000),
    lov_null_value                   VARCHAR2(4000),
    lov_cascade_parent_items        VARCHAR2(4000),
    lov_return_column                VARCHAR2(128),
    lov_display_column               VARCHAR2(128),
    lov_icon_column                 VARCHAR2(128),
    lov_group_column                 VARCHAR2(128),
    lov_group_sort_direction        VARCHAR2(16),
    lov_default_sort_column         VARCHAR2(128),
    lov_default_sort_direction     VARCHAR2(16),
    lov_oracle_text_column          VARCHAR2(128),
    lov_columns                      t_lov_columns,
    lov_is_legacy                    BOOLEAN,
    ajax_items_to_submit             VARCHAR2(4000),
    ajax_optimize_refresh           BOOLEAN,
    element_width                   NUMBER,
    element_max_length              NUMBER,
    element_height                  NUMBER,
    element_css_classes             VARCHAR2(4000),
    element_attributes              VARCHAR2(4000),
    element_option_attributes       VARCHAR2(4000),
    icon_css_classes                VARCHAR2(4000),
    escape_output                    BOOLEAN,
    ignore_change                   BOOLEAN DEFAULT TRUE,
    --
)
```

```

        attributes          apex_t_plugin_attributes,
        --
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),
init_javascript_code VARCHAR2(32767),
inline_help_text      VARCHAR2(4000),
inline_help_id        VARCHAR2(270),
);

```

44.3.18 t_item_ajax_result

The `t_item_ajax_result` type is used as the result type for the Ajax function of an item type plug-in.

```

type t_item_ajax_result is record (
    dummy boolean /* not used yet */
);

```

44.3.19 t_item_meta_data_result

The `t_item_meta_data_result` type is used as the result type for the meta data function of an item type plug-in.

Syntax

```

TYPE T_ITEM_META_DATA_RESULT IS RECORD (
    is_multi_value      BOOLEAN DEFAULT FALSE, /* (Deprecated) Declare
if multiple values can be selected
                                                in an LOV-based item plug-
in */
    display_lov_definition VARCHAR2(32767),      /* Provides the lov

```

```

definition (SQL-statement) to the
                                interactive grid */
    return_display_value      BOOLEAN DEFAULT TRUE, /* Declare if item plug-
in has a display and return
                                value or just a return
value */
    escape_output              BOOLEAN DEFAULT TRUE, /* Declare if output
should be escaped or not e.g. in
                                Interactive Grid. Used
for HTML Markup based items
                                like an image item plug-
in */
    container_css_classes     VARCHAR2(32767)        /* Add CSS classes on
container level for an item plug-in */
);

```

44.3.20 t_item_render_param

The `t_item_render_param` type is passed into render procedure of the item type plug-in and contains information about the current page item value.

```

type t_item_render_param is record (
    value_set_by_controller BOOLEAN DEFAULT FALSE,
    value                  VARCHAR2(32767),
    clob_value             CLOB,
    is_READONLY            BOOLEAN DEFAULT FALSE,
    is_printer_friendly   BOOLEAN DEFAULT FALSE
);

```

44.3.21 t_item_render_result

The `t_item_render_result` type is used as the result type for the rendering function of an item type plug-in.

```

type t_item_render_result is record (
    is_navigable           boolean default false,
    navigable_dom_id       varchar2(255),          /* should only be set if
navigable element is not equal to item name */
    item_rendered          boolean default true   /* should be set to false
if the render procedure didn't render anything,
                                this could be the case
for a read only item in IG */
);

```

44.3.22 t_item_validation_result

The `t_item_validation_result` type is used as the result type for the validation function of an item type plug-in.

```

type t_item_validation_result is record (
    message                varchar2(32767),
    display_location        varchar2(40),          /* if not set the application default
is used */

```

```
    page_item_name  varchar2(255) ); /* if not set the validated page item
name is used */
```

44.3.23 t_plugin

The `t_plugin` type is passed into all plug-in functions and contains information about the current plug-in.

```
type t_plugin is record (
    name          varchar2(45),
    file_prefix   varchar2(4000),
    attributes    t_plugin_attributes, /* used by region and item plug-ins */
    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) );
```

44.3.24 t_plugin_attributes

```
type t_plugin_attributes is object (
    function get_varchar2 (
        p_static_id      IN VARCHAR2
        --
        p_default_value   IN VARCHAR2      DEFAULT NULL,
        p_do_substitutions IN BOOLEAN     DEFAULT FALSE,
        p_do_serveronly_substitutions IN BOOLEAN     DEFAULT FALSE,
        p_substitutions_escape_mode   IN apex_session_state.t_escape_mode
                                         DEFAULT
                                         apex_session_state.c_escape_mode_html )
        RETURN VARCHAR2
        --
        function get_number (
            p_static_id      IN VARCHAR2,
            --
            p_default_value IN NUMBER DEFAULT NULL )
        RETURN NUMBER

        --Y returns TRUE / N returns FALSE
        function get_boolean (
            p_static_id      IN VARCHAR2,
            --
            p_default_value IN BOOLEAN DEFAULT NULL )
```

```
    RETURN BOOLEAN  
);
```

 **Note:**

The parameters `p_do_substitutions` and `p_do_serveronly_substitutions` **cannot** be used in the same function call.

44.3.25 t_process

The `t_process` type is passed into all process type plug-in functions and contains information about the current process.

```
type t_process is record (  
    id             number,  
    name           varchar2(255),  
    region_id      number,  
    row_num        number,  
    correlation_context  varchar2(4000),  
    component_type  varchar2(30),  
    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) );
```

44.3.26 t_process_exec_result

The `t_process_exec_result` type is used as the result type for the execution function of a process type plug-in.

```
type t_process_exec_result is record (  
    success_message varchar2(32767)  
    execution_skipped boolean default false /* set to TRUE if process  
execution has been skipped by plug-in because of additional condition checks  
*/  
);
```

44.3.27 t_region

The `t_region` 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(4000),
    title                            VARCHAR2(4000),
    type                             VARCHAR2(255),
    source                           VARCHAR2(32767),
    lazy_loading                     BOOLEAN,
    ajax_items_to_submit              VARCHAR2(32767),
    ajax_items_to_submit_singlerow    VARCHAR2(32767),
    fetched_rows                     PLS_INTEGER,
    escape_output                     BOOLEAN,
    error_message                    VARCHAR2(32767), /* obsolete */
    no_data_found_message            VARCHAR2(32767),
    attributes                        t_plugin_attributes, /* only used by
region plug-ins */
    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),
    filter_region_id                 NUMBER,
    filter_region_static_id           VARCHAR2(255),
    region_columns                   t_region_columns,
    init_javascript_code              VARCHAR2(32767) );
```

44.3.28 t_region_ajax_result

The `t_region_ajax_result` type is used as result type for the Ajax function of a region type plug-in.

```
type t_region_ajax_result is record (
    dummy boolean /* not used yet */
);
```

44.3.29 t_region_column

The `t_region_column` 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,
    attributes              t_plugin_attributes,
    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);
```

44.3.30 t_region_columns

```
type t_region_columns is table of t_region_column index by
    pls_integer;
```

44.3.31 t_region_render_param

```
type t_region_render_param is record (
    is_printer_friendly boolean
);
```

44.3.32 t_region_render_result

The `t_region_render_result` type is used as the result type for the rendering function of a region type plug-in.

```
type t_region_render_result is record (
    navigable_dom_id varchar2(255) /* can be used to put focus to an input
        field (that is, search field) the region renders as part of the plug-in
        output */
);
```

44.3.33 t_remote_server_config

Used in the configuration procedure of flexible servers.

```
type t_remote_server_config is record(
    base_url          varchar2(4000),
    substitutions     wwv_flow_t_varchar2 );
```

44.3.34 t_remote_server_info

Used in the configuration procedure of flexible servers.

```
type t_remote_server_info is record(
    application_id      number,
    remote_server_static_id varchar2(255) );
```

44.4 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;
```

44.5 GET_INPUT_NAME_FOR_PAGE_ITEM Function (Deprecated)

 **Caution:**

This API is deprecated and will be removed in a future release.

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 APEX can map the submitted value to the actual page item in session state. This function returns the mapping `name` for your page item.

 **Note:**

This function is only useful when called in the rendering function of an item type plug-in.

Syntax

```
FUNCTION GET_INPUT_NAME_FOR_PAGE_ITEM (
    P_IS_MULTI_VALUE IN BOOLEAN )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_is_multi_value	If the HTML input element has multiple values, such as a select list with <code>multiple="multiple"</code> , then set p_is_multi_value to TRUE.

Example

The following example outputs the necessary HTML code to render a text field where the value gets stored in session state when the page is submitted.

```
sys.htp.prn (
    '<input type="text" id="'||p_item.name||'" ' ||
    'name="'||apex_plugin.get_input_name_for_page_item(false)||'" ' ||
    'value="'||sys.htf.escape_sc(p_value)||'" ' ||
    'size="'||p_item.element_width||'" ' ||
    'maxlength="'||p_item.element_max_length||'" ' ||
    coalesce(p_item.element_attributes, 'class="text_field"')||' />' );
```

APEX_PLUGIN_UTIL

The `APEX_PLUGIN_UTIL` package provides utility functions that solve common problems when writing a plug-in.

- [BUILD_REQUEST_BODY](#) Procedure
- [CLEAR_COMPONENT_VALUES](#) Procedure
- [CURRENT_ROW_CHANGED](#) Function
- [DB_OPERATION_ALLOWED](#) Function
- [DEBUG_DYNAMIC_ACTION](#) Procedure
- [DEBUG_PAGE_ITEM](#) Procedure Signature 1 (Deprecated)
- [DEBUG_PAGE_ITEM](#) Procedure Signature 2 (Deprecated)
- [DEBUG_PROCESS](#) Procedure
- [DEBUG_REGION](#) Procedure Signature 1
- [DEBUG_REGION](#) Procedure Signature 2
- [ESCAPE](#) Function
- [EXECUTE_PLSQL_CODE](#) Procedure (Deprecated)
- [GET_ATTRIBUTE_AS_NUMBER](#) Function
- [GET_CURRENT_DATABASE_TYPE](#) 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_HTML_ATTR](#) Function
- [GET_ORDERBY_NULLS_SUPPORT](#) Function
- [GET_PLSQL_EXPR_RESULT_BOOLEAN](#) Function
- [GET_PLSQL_EXPR_RESULT_CLOB](#) Function
- [GET_PLSQL_EXPRESSION_RESULT](#) Function
- [GET_PLSQL_FUNC_RESULT_BOOLEAN](#) Function
- [GET_PLSQL_FUNC_RESULT_CLOB](#) Function
- [GET_PLSQL_FUNCTION_RESULT](#) Function
- [GET_POSITION_IN_LIST](#) Function (Deprecated)
- [GET_SEARCH_STRING](#) Function

- GET_VALUE_AS_VARCHAR2 Function
- GET_WEB_SOURCE_OPERATION Function
- IS_EQUAL Function
- IS_COMPONENT_USED Function
- MAKE_REST_REQUEST Procedure Signature 1
- MAKE_REST_REQUEST Procedure Signature 2
- PAGE_ITEM_NAMES_TO_JQUERY Function (Deprecated)
- PARSE_REFETCH_RESPONSE Function
- PRINT_DISPLAY_ONLY Procedure Signature 1 (Deprecated)
- PRINT_DISPLAY_ONLY Procedure Signature 2 (Deprecated)
- PRINT_ESCAPED_VALUE Procedure Signature 1
- PRINT_ESCAPED_VALUE Procedure Signature 2
- PRINT_HIDDEN Procedure
- PRINT_HIDDEN_IF_READONLY Procedure
- PRINT_JSON_HTTP_HEADER Procedure
- PRINT_LOV_AS_JSON Procedure
- PRINT_OPTION Procedure
- PRINT_READ_ONLY Procedure Signature 1
- PRINT_READ_ONLY Procedure Signature 2
- PROCESS_DML_RESPONSE Procedure
- REPLACE_SUBSTITUTIONS Function
- SET_COMPONENT_VALUES Procedure
- SPLIT_MULTIPLE_VALUE_TO_TABLE Function

45.1 BUILD_REQUEST_BODY Procedure

This procedure builds a request body for a REST Data Source DML request. If a request body template is set, then #COLUMN# placeholders will be replaced by the DML context column values. In this case, the request body can be any data format.

If no request body template is set, the function builds a JSON with the following structure:

```
{  
  "{column1-name}": "{column1-value}",  
  "{column2-name}": "{column2-value}",  
  :  
}
```

Syntax

```
APEX_PLUGIN_UTIL.BUILD_REQUEST_BODY (  
  p_request_format    IN          apex_plugin.t_data_format,  
  p_profile_columns   IN          apex_plugin.t_web_source_columns,  
  p_values_context    IN          apex_exec.t_context,
```

```

    p_build_when_empty   IN          BOOLEAN,
    --
    p_request_body       IN OUT NOCOPY CLOB );

```

Parameters

Parameter	Description
p_request_format	Request format (JSON or XML).
p_profile_columns	Column meta data (names, data types).
p_values_context	apex_exec context object containing DML values.
p_build_when_empty	If p_request_body is empty, whether to build a new request body.
p_request_body	Request body template to perform replacements on.

Returns

Parameter	Description
p_request_body	Request body (substitutions replaced or built from scratch).

Example

The following example uses BUILD_REQUEST_BODY within a plug-in DML procedure.

```

procedure plugin_dml(
    p_plugin      IN          apex_plugin.t_plugin,
    p_web_source  IN          apex_plugin.t_web_source,
    p_params      IN          apex_plugin.t_web_source_dml_params,
    p_result      IN OUT NOCOPY apex_plugin.t_web_source_dml_result )
IS
    l_web_source_operation apex_plugin.t_web_source_operation;
    l_request_body         clob;
BEGIN

    l_web_source_operation := apex_plugin_util.get_web_source_operation(
        p_web_source  => p_web_source,
        p_db_operation => apex_plugin.c_db_operation_insert,
        p_perform_init => true );

    apex_plugin_util.build_request_body(
        p_request_format      => apex_plugin.c_format_json,
        p_profile_columns     => p_web_source.profile_columns,
        p_values_context      => p_params.insert_values_context,
        p_build_when_empty    => true,
        p_request_body        => l_request_body );

    -- continue with APEX_PLUGIN_UTIL.MAKE_REST_REQUEST

END plugin_dml;

```

45.2 CLEAR_COMPONENT_VALUES Procedure

This procedure clears the component specific Session State set by apex_plugin_util.set_component_values.

Syntax

```
procedure clear_component_values;
```

Example

See apex_plugin_util.set_component_values

See Also:

[SET_COMPONENT_VALUES Procedure](#)

45.3 CURRENT_ROW_CHANGED Function

This function determines whether the current row changed between the two contexts. In order to compare the next row within the value context, use APEX_EXEC.NEXT_ROW for both contexts.

Syntax

```
API_PLUGIN_UTIL.CURRENT_ROW_CHANGED(
    p_old_row_context      IN apex_exec.t_context,
    p_new_row_context      IN apex_exec.t_context )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_old_row_context	Values context containing values before the change.
p_new_row_context	Values context containing values after the change.

Returns

Parameter	Description
*	Whether there is a difference between the rows.

Example

The following example performs a "refetch" operation within the Plug-In DML function for a given row to be updated and check whether the row would actually be changed with the DML operation. If not, we could suppress the HTTP request.

```

procedure plugin_dml(
    p_plugin      in      apex_plugin.t_plugin,
    p_web_source  in      apex_plugin.t_web_source,
    p_params      in      apex_plugin.t_web_source_dml_params,
    p_result      in out nocopy apex_plugin.t_web_source_dml_result )
IS
    l_web_source_operation apex_plugin.t_web_source_operation;
    l_request_body         clob;
    l_response             clob;

    l_refetch_context      apex_exec.t_context;
    l_checksum              varchar2(32767);
    l_refetched_checksum   varchar2(32767);

BEGIN
    p_result.update_values_context := p_params.update_values_context;

    --
    -- this code performs a "refetch" operation for a row, in order to perform
    -- lost update detection. This happens before the actual DML.
    --
    IF p_web_source.operations.exists( apex_plugin.c_db_operation_fetch_row )
THEN

    l_web_source_operation := apex_plugin_util.get_web_source_operation(
        p_web_source      => p_web_source,
        p_db_operation    => apex_plugin.c_db_operation_fetch_row,
        p_preserve_headers => false,
        p_perform_init     => true );

    -- add some logic to add primary key values to the URL or as HTTP
    headers here
    -- PK values can be obtained from "p_params.update_values_context"

    apex_plugin_util.make_rest_request(
        p_web_source_operation => l_web_source_operation,
        p_request_body        => l_request_body,
        p_response            => l_response,
        p_response_parameters => p_result.out_parameters );

    l_refetch_context := apex_plugin_util.parse_refetch_response(
        p_web_source_operation => l_web_source_operation,
        p_web_source           => p_web_source,
        p_response             => l_response,
        p_values_context       => p_params.update_values_context );

    IF apex_plugin_util.current_row_changed(
        p_old_row_context    => l_refetch_context,
        p_new_row_context    => p_params.update_values_context )

```

```

    THEN
        -- perform actual DML here
        --
    ELSE
        apex_exec.set_row_status(
            p_context  => p_result.update_values_context,
            p_sqlcode   => 0,
            p_sqlerrm  => 'SKIPPED' );
    END IF;
END IF;
END plugin_dml;

```

45.4 DB_OPERATION_ALLOWED Function

This function checks whether a database operation is allowed (contained in the allowed operations) and either raises an APEX error or returns an error message.

Syntax

```

APEX_PLUGIN_UTIL.DB_OPERATION_ALLOWED (
    p_allowed_operations    IN VARCHAR2,
    p_operation              IN apex_plugin.t_db_operation,
    p_raise_error            IN BOOLEAN DEFAULT TRUE )
RETURN VARCHAR2;

```

Parameters

Parameter	Description
p_allowed_operations	Allowed operations (U, UD, D).
p_operation	Operation to check for.
p_raise_error	Whether to raise an error if the operation is not allowed (default TRUE).

Returns

NULL if the operation is allowed.

If not allowed, an error message and p_raise_error is FALSE.

Example

The following example asserts (using allowed_operations_column) that the current operation is allowed within the Plug-In code.

```

DECLARE
    l_error_message varchar2(32767);
BEGIN
    l_error_message := apex_plugin_util.db_operation_allowed(
        p_allowed_operations => apex_exec.get_varchar2(
            p_context      =>
            l_refetch_context,
            p_column_name  =>
            p_params.allowed_operations_column ),

```

```

          p_operation      =>
apex_plugin.c_db_operation_update,
          p_raise_error    => false );
END;

```

45.5 DEBUG_DYNAMIC_ACTION Procedure

This procedure writes the data of the dynamic action meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_DYNAMIC_ACTION (
    p_plugin      IN apex_plugin.t_plugin,
    p_dynamic_action IN apex_plugin.t_dynamic_action );
```

Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_dynamic_action	This is the p_dynamic_action parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the rendered function or Ajax callback function of the plug-in.

```
apex_plugin_util.debug_dynamic_action (
    p_plugin      => p_plugin,
    p_dynamic_action => p_dynamic_action );
```

45.6 DEBUG_PAGE_ITEM Procedure Signature 1 (Deprecated)



Note:

This API is deprecated and will be removed in a future release.

This procedure writes the data of the page item meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_PAGE_ITEM (
    p_plugin      IN apex_plugin.t_plugin,
    p_page_item   IN apex_plugin.t_page_item );
```

Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_page_item	This is the p_page_item parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the renderer, Ajax callback or validation function.

```
apex_plugin_util.debug_page_item (
    p_plugin      => p_plugin,
    p_page_item   => p_page_item );
```

45.7 DEBUG_PAGE_ITEM Procedure Signature 2 (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

This procedure writes the data of the page item meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_PAGE_ITEM (
    p_plugin          IN apex_plugin.t_plugin,
    p_page_item       IN apex_plugin.t_page_item,
    p_value           IN VARCHAR2,
    p_is_READONLY     IN BOOLEAN,
    p_is_printer_FRIENDLY IN BOOLEAN );
```

Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_page_item	This is the p_page_item parameter of your plug-in function.
p_value	This is the p_value parameter of your plug-in function.
p_is_READONLY	This is the p_is_READONLY parameter of your plug-in function.
p_is_printer_FRIENDLY	This is the p_is_printer_FRIENDLY parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the renderer, Ajax callback or validation function.

```
apex_plugin_util.debug_page_item (
    p_plugin      => p_plugin,
    p_page_item   => p_page_item,
    p_value       => p_value,
    p_is_READONLY => p_is_READONLY,
    p_is_printer_FRIENDLY => p_is_printer_FRIENDLY);
```

45.8 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

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_process	This is the p_process parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the execution function of the plug-in.

```
apex_plugin_util.debug_process (
    p_plugin      => p_plugin,
    p_process     => p_process);
```

45.9 DEBUG_REGION Procedure Signature 1

This procedure writes the data of the region meta data to the debug output if debugging is enabled.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_REGION (
    p_plugin      IN apex_plugin.t_plugin,
    p_region      IN apex_plugin.t_region );
```

Parameters

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_region	This is the p_region parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the render function or Ajax callback function of the plug-in.

```
apex_plugin_util.debug_region (
    p_plugin      => p_plugin,
    p_region      => p_region);
```

45.10 DEBUG_REGION Procedure Signature 2

This procedure writes the data of the region meta data to the debug output if debugging is enabled. This is the advanced version of the debugging procedure which is used for the rendering function of a region plug-in.

Syntax

```
APEX_PLUGIN_UTIL.DEBUG_REGION (
    p_plugin      IN apex_plugin.t_plugin,
    p_region      IN apex_plugin.t_region,
    p_is_printer_friendly IN BOOLEAN );
```

Parameters

Parameters for procedure.

Parameter	Description
p_plugin	This is the p_plugin parameter of your plug-in function.
p_region	This is the p_region parameter of your plug-in function.
p_is_printer_friendly	This is the p_is_printer_friendly parameter of your plug-in function.

Example

This example shows how to collect helpful debug information during the plug-in development cycle to see what values are actually passed into the render function or Ajax callback function of the plug-in.

```
apex_plugin_util.debug_region (
    p_plugin      => p_plugin,
    p_region      => p_region,
    p_is_printer_friendly => p_is_printer_friendly);
```

45.11 ESCAPE Function

This function is used when the standard attribute "Has Escape Output Attribute" option is enabled for an item type plug-in which enables a developer to choose whether to escape the output.

Syntax

```
APEX_PLUGIN_UTIL.ESCAPE (
    p_value      IN VARCHAR2,
    p_escape     IN BOOLEAN )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	This is the value you want to escape depending on the p_escape parameter.
p_escape	If TRUE, the return value is escaped. If FALSE, the value is not escaped.

Example

This example outputs all values of the array l_display_value_list as a HTML list and escapes the value of the array depending on the setting the developer chose when using the plug-in.

```
FOR i IN 1 .. l_display_value_list.count
LOOP
    sys.htp.prn (
        '<li>' ||
        apex_plugin_util.escape (
            p_value  => l_display_value_list(i),
            p_escape => p_item.escape_output ) ||
        '</li>');
END LOOP;
```

45.12 EXECUTE_PLSQL_CODE Procedure (Deprecated)



Note:

This API is deprecated and will be removed in a future release.

Use [EXECUTE_PLSQL Procedure Signature 1](#) or [EXECUTE_PLSQL Procedure Signature 2](#) instead.

This procedure executes a PL/SQL code block and performs binding of bind variables in the provided PL/SQL code.

Syntax

```
APEX_PLUGIN_UTIL.EXECUTE_PLSQL_CODE (
    p_plsql_code      IN VARCHAR2,
    p_auto_bind_items IN BOOLEAN      DEFAULT TRUE,
    p_bind_list       IN t_bind_list DEFAULT c_empty_bind_list );
```

Parameters

Parameter	Description
p_plsql_code	PL/SQL code to be executed.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Example

Text which should be escaped and then printed to the HTTP buffer.

```
DECLARE
    l_plsql_code VARCHAR2 := p_process.attribute_01;
BEGIN
    apex_plugin_util.execute_plsql_code (
        p_plsql_code => l_plsql_code );
END;
```

45.13 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

Parameter	Description
p_attribute_label	The label of the custom plug-in attribute.
p_value	The value of a custom attribute of type NUMBER.

Example

```
declare
    l_value number;
begin
    -- The following may fail for languages that don't use dot as the NLS
```

```

decimal separator
    l_value := to_number( p_region.attribute_04 );

    -- The following will work correctly regardless of the effective NLS
    decimal separator
        l_value :=
apex_plugin_util.get_attribute_as_number( p_region.attribute_04, 'Minimum
Amount' );
end;
/

```

45.14 GET_CURRENT_DATABASE_TYPE Function

This function retrieves the database type for the currently active region. If Plug-In developers generate SQL in their code, this information helps to generate correct SQL for the corresponding database type.

Syntax

```
APEX_PLUGIN_UTIL.GET_CURRENT_DATABASE_TYPE (
    p_remote_server_id    IN NUMBER    DEFAULT NULL )
RETURN apex_exec.t_database_type;
```

Parameters

Parameter	Description
p_remote_server_id	The internal ID of the REST Enabled SQL reference.

Returns

This function returns the database vendor for the data source of the currently executed region.

Example

```

DECLARE
    l_database_type apex_exec.t_database_type;
BEGIN
    l_database_type := apex_plugin_util.get_current_database_type;
    IF l_database_type = apex_exec.c_database_mysql THEN
        -- MySQL specific code goes here
    ELSE
        -- normal code goes here
    END IF;
END;
```

45.15 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,
    p_search_column_no   IN VARCHAR2,
    p_search_string       IN VARCHAR2,
    p_first_row           IN NUMBER      DEFAULT NULL,
    p_max_rows             IN NUMBER      DEFAULT NULL,
    p_auto_bind_items     IN BOOLEAN     DEFAULT TRUE,
    p_bind_list            IN t_bind_list DEFAULT c_empty_bind_list,
    RETURN t_column_value_list;
```

Parameters

Parameters	Description
p_sql_statement	SQL statement used for the lookup.
p_min_columns	Minimum number of return columns.
p_max_columns	Maximum number of return columns.
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_search_type	Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns through p_max_columns range.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
t_column_value_list	Table of apex_application_global.vc_arr2 indexed by column number.

Example

The following example shows a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list.

```
function render_list (
    p_item          in apex_plugin.t_page_item,
```

```

    p_value          in varchar2,
    p_is_READONLY    in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_column_value_list  apex_plugin_util.t_column_value_list;
begin
    l_column_value_list :=
        apex_plugin_util.get_data (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
            p_component_name   => p_item.name,
            p_search_type       => apex_plugin_util.c_search_contains_case,
            p_search_column_no => 1,
            p_search_string     => p_value );

    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list(1).count
    loop
        sys.htp.p(
            '<li>' ||
            sys.htf.escape_sc(l_column_value_list(1)(i)) || -- display column
            '-' ||
            sys.htf.escape_sc(l_column_value_list(2)(i)) || -- return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;

```

45.16 GET_DATA Function Signature 2

Executes the specified SQL query restricted by the provided search string (optional) and returns the values for each column. All column values are returned as a string, independent of their data types. The search column is identified by providing a column name in the `p_search_column_name` parameter. This function takes into account character value comparison globalization attributes defined for the application.

Syntax

```

APEX_PLUGIN_UTIL.GET_DATA (
    p_sql_statement      IN VARCHAR2,
    p_min_columns        IN NUMBER,
    p_max_columns        IN NUMBER,
    p_component_name     IN VARCHAR2,
    p_search_type         IN VARCHAR2 DEFAULT NULL,
    p_search_column_name IN VARCHAR2 DEFAULT NULL,
    p_search_string       IN VARCHAR2 DEFAULT NULL,
    p_first_row           IN NUMBER DEFAULT NULL,
    p_max_rows             IN NUMBER DEFAULT NULL,
    p_auto_bind_items     IN BOOLEAN DEFAULT TRUE,
    p_bind_list            IN t_bind_list DEFAULT c_empty_bind_list )
RETURN t_column_value_list;

```

Parameters

Parameters	Description
p_sql_statement	SQL statement used for the lookup.
p_min_columns	Minimum number of return columns.
p_max_columns	Maximum number of return columns.
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_search_type	Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore
p_search_column_name	This is the column name used to restrict the SQL statement.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Parameter	Description
t_column_value_list	Table of apex_application_global.vc_arr2 indexed by column number.

Example

The following example shows a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list.

```

function render_list (
    p_item          in apex_plugin.t_page_item,
    p_value         in varchar2,
    p_is_READONLY   in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_column_value_list    apex_plugin_util.t_column_value_list;
begin
    l_column_value_list :=
        apex_plugin_util.get_data (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
            p_component_name   => p_item.name,
            p_search_type       => apex_plugin_util.c_search_contains_case,
            p_search_column_name => 'ENAME',
            p_search_string     => p_value );
    sys.htp.p('<ul>');

```

```

        for i in 1 .. l_column_value_list(1).count
    loop
        sys.htp.p(
            '<li>' ||
            sys.htf.escape_sc(l_column_value_list(1)(i)) || -- display column
            '-' ||
            sys.htf.escape_sc(l_column_value_list(2)(i)) || -- return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;

```

45.17 GET_DATA2 Function Signature 1

This function executes the specified SQL query (optionally restricted by the provided search string) 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 apex_application_global.vc_arr2 DEFAULT
c_empty_data_type_list,
    p_component_name     IN VARCHAR2,
    p_search_type         IN VARCHAR2,
    p_search_column_no   IN VARCHAR2,
    p_search_string       IN VARCHAR2,
    p_first_row           IN NUMBER                      DEFAULT NULL,
    p_max_rows             IN NUMBER                     DEFAULT NULL,
    p_auto_bind_items     IN BOOLEAN                    DEFAULT TRUE,
    p_bind_list            IN t_bind_list               DEFAULT
c_empty_bind_list )
    RETURN t_column_list;

```

Parameters

Parameter	Description
<code>p_sql_statement</code>	SQL statement used for the lookup.
<code>p_min_columns</code>	Minimum number of return columns.
<code>p_max_columns</code>	Maximum number of return columns.
<code>p_data_type_list</code>	If provided, checks to make sure the data type for each column matches the specified data type in the array. Use the constants <code>c_data_type_*</code> for available data types.
<code>p_component_name</code>	In case an error is returned, this is the name of the page item or report column used to display the error message.

Parameter	Description
p_search_type	Must be one of the following c_search_* constants: <ul style="list-style-type: none">• c_search_contains_case• c_search_contains_ignore• c_search_exact_case• c_search_exact_ignore
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns through p_max_columns range.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
t_column_list	Table of t_column_values indexed by column number.

Example 1

In the following example, a simple item type plug-in rendering function executes the LOV defined for the page item and performs a case sensitive LIKE filtering with the current value of the page item. The result then generates as an HTML list. Here, the first column of the LOV SQL statement is checked if it is VARCHAR2 and the second is NUMBER.

```

function render_list (
    p_item          in apex_plugin.t_page_item,
    p_value         in varchar2,
    p_is_READONLY   in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
IS
    l_data_type_list    apex_application_global.vc_arr2;
    l_column_value_list apex_plugin_util.t_column_value_list2;
BEGIN
    -- The first LOV column has to be a string and the second a number
    l_data_type_list(1) := apex_plugin_util.c_data_type_varchar2;
    l_data_type_list(2) := apex_plugin_util.c_data_type_number;
    --
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement    => p_item.lov_definition,
            p_min_columns      => 2,
            p_max_columns      => 2,
            p_data_type_list   => l_data_type_list,
            p_component_name   => p_item.name,
            p_search_type       => apex_plugin_util.c_search_contains_case,
            p_search_column_no  => 1,
            p_search_string     => p_value );
    --

```

```

        sys.htp.p('<ul>');
        FOR i in 1 .. l_column_value_list.count
        LOOP
            sys.htp.p(
                '<li>' ||
                sys.hf.escape_sc(l_column_value_list(1).value_list(i).varchar2_value)|| --
                display column
                '-'||

                sys.hf.escape_sc(l_column_value_list(2).value_list(i).number_value)|| --
                return column
                '</li>');
            END LOOP;
            sys.htp.p('</ul>');
        END render_list;
    
```

Example 2

In the following example, a simple region type plug-in rendering function executes the SQL query defined for the region. The result generates as an HTML list. This example demonstrates the advanced handling of object type columns like SDO_Geometry.

```

function render (
    p_region in apex_plugin.t_region,
    p_plugin in apex_plugin.t_plugin,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_region_render_result
IS
    l_column_value_list apex_plugin_util.t_column_value_list2;
    l_geometry sdo_geometry;
    l_value varchar2(32767);
    l_dummy pls_integer;
BEGIN
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement => p_region.source,
            p_min_columns => 1,
            p_max_columns => null,
            p_component_name => p_region.name );
    --
    sys.htp.p('<ul>');
    FOR row in 1 .. l_column_value_list(1).value_list.count LOOP
        sys.htp.p('<li>');
        FOR col in 1 .. l_column_value_list.count LOOP
            IF l_column_value_list(col).data_type = 'SDO_GEOGRAPHY' 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 );
            END IF;
        END LOOP;
    END LOOP;
END;
    
```

```

        l_value := '(' type=' || l_geometry.sdo_gtype || ' srid=' ||
l_geometry.sdo_srid ||
        case when l_geometry.sdo_point is not null THEN
            ',x=' || l_geometry.sdo_point.x ||
            ',y=' || l_geometry.sdo_point.y ||
            ',z=' || l_geometry.sdo_point.z
        END ||
        ')';
    ELSE
        l_value := apex_plugin_util.get_value_as_varchar2(
            p_data_type =>
l_column_value_list(col).data_type,
            p_value =>
l_column_value_list(col).value_list(row) );
        END IF;

        sys.htp.p( case when col > 1 then ' - ' END || l_value );
    END LOOP;

    sys.htp.p('<li>');
END LOOP;
sys.htp.p('<ul>');

RETURN null;
END;

```

45.18 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 name in the `p_search_column_name` 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 apex_application_global.vc_arr2 DEFAULT
c_empty_data_type_list,
    p_component_name    IN VARCHAR2,
    p_search_type        IN VARCHAR2                               DEFAULT NULL,
    p_search_column_name IN VARCHAR2                               DEFAULT NULL,
    p_search_string      IN VARCHAR2                               DEFAULT NULL,
    p_first_row          IN NUMBER                               DEFAULT NULL,
    p_max_rows           IN NUMBER                               DEFAULT NULL,
    p_auto_bind_items   IN BOOLEAN                             DEFAULT TRUE,
    p_bind_list          IN t_bind_list                         DEFAULT
c_empty_bind_list )
    RETURN t_column_list;

```

Parameters

Parameter	Description
p_sql_statement	SQL statement used for the lookup.
p_min_columns	Minimum number of return columns.
p_max_columns	Maximum number of return columns.
p_data_type_list	If provided, checks to make sure the data type for each column matches the specified data type in the array. Use the constants c_data_type_* for available data types.
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_search_type	Must be one of the c_search_* constants. They are as follows: c_search_contains_case, c_search_contains_ignore, c_search_exact_case, c_search_exact_ignore
p_search_column_name	The column name used to restrict the SQL statement.
p_search_string	Value used to restrict the query.
p_first_row	Start query at the specified row. All rows before the specified row are skipped.
p_max_rows	Maximum number of return rows allowed.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Parameter	Description
t_column_list	Table of t_column_values indexed by column number.

Example

The following example is a simple item type plug-in rendering function which executes the LOV defined for the page item and does a case sensitive LIKE filtering with the current value of the page item. The result is then generated as a HTML list. This time, the first column of the LOV SQL statement is checked if it is of type VARCHAR2 and the second is of type number.

```

function render_list (
    p_item          in apex_plugin.t_page_item,
    p_value         in varchar2,
    p_is_READONLY   in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_data_type_list    apex_application_global.vc_arr2;
    l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    -- The first LOV column has to be a string and the second a number
    l_data_type_list(1) := apex_plugin_util.c_data_type_varchar2;
    l_data_type_list(2) := apex_plugin_util.c_data_type_number;
    --
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement    => p_item.lov_definition,

```

```

        p_min_columns      => 2,
        p_max_columns      => 2,
        p_data_type_list   => l_data_type_list,
        p_component_name   => p_item.name,
        p_search_type       => apex_plugin_util.c_search_contains_case,
        p_search_column_name => 'ENAME',
        p_search_string     => p_value );
    --
    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list.count(1)
    loop
        sys.htp.p(
            '<li>' ||
            sys.hf.escape_sc(l_column_value_list(1).value_list(i).varchar2_value)|| --
            display column
            ' -' ||

            sys.hf.escape_sc(l_column_value_list(2).value_list(i).number_value)|| --
            return column
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;

```

45.19 GET_DISPLAY_DATA Function Signature 1

This function gets the display lookup value for the value specified in `p_search_string`.

Syntax

```

APEX_PLUGIN_UTIL.GET_DISPLAY_DATA (
    p_sql_statement      IN VARCHAR2,
    p_min_columns        IN NUMBER,
    p_max_columns        IN NUMBER,
    p_component_name     IN VARCHAR2,
    p_display_column_no  IN BINARY_INTEGER DEFAULT 1,
    p_search_column_no   IN BINARY_INTEGER DEFAULT 2,
    p_search_string       IN VARCHAR2          DEFAULT NULL,
    p_display_extra       IN BOOLEAN           DEFAULT TRUE,
    p_escape_display_extra IN BOOLEAN           DEFAULT TRUE,
    p_auto_bind_items    IN BOOLEAN           DEFAULT TRUE,
    p_bind_list           IN t_bind_list        DEFAULT c_empty_bind_list,
    RETURN VARCHAR2;

```

Parameters

Parameter	Description
<code>p_sql_statement</code>	SQL statement used for the lookup.
<code>p_min_columns</code>	Minimum number of return columns.
<code>p_max_columns</code>	Maximum number of return columns.

Parameter	Description
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_display_column_no	Number of the column returned from the SQL statement. Must be within the p_min_columns through p_max_columns range.
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns through p_max_columns range.
p_search_string	Value to look up.
p_display_extra	If set to TRUE, and a value is not found, the search value is added to the result instead.
p_escape_display_extra	If TRUE, p_search_string is escaped if added as "Display Extra" value.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
VARCHAR2	Value of the first record of the column specified by p_display_column_no. If no record was found it contains the value of p_search_string if the parameter p_display_extra is set to TRUE. Otherwise NULL is returned.

Example

The following example does a lookup with the value provided in p_value and returns the display column of the LOV query.

```
function render_value (
    p_item          in apex_plugin.t_page_item,
    p_value         in varchar2,
    p_is_READONLY   in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
begin
    sys.htp.p(sys.htf.escape_sc(
        apex_plugin_util.get_display_data (
            p_sql_statement      => p_item.lov_definition,
            p_min_columns        => 2,
            p_max_columns        => 2,
            p_component_name     => p_item.name,
            p_display_column_no  => 1,
            p_search_column_no   => 2,
            p_search_string       => p_value )) );
end render_value;
```

45.20 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 apex_application_global_vc_arr2,
    p_display_extra       IN BOOLEAN      DEFAULT TRUE,
    p_escape_display_extra IN BOOLEAN      DEFAULT TRUE,
    p_auto_bind_items    IN BOOLEAN      DEFAULT TRUE,
    p_bind_list           IN t_bind_list  DEFAULT c_empty_bind_list,
    RETURN apex_application_global_vc_arr2;
```

Parameters

Parameter	Description
p_sql_statement	SQL statement used for the lookup.
p_min_columns	Minimum number of return columns.
p_max_columns	Maximum number of return columns.
p_component_name	In case an error is returned, this is the name of the page item or report column used to display the error message.
p_display_column_no	Number of the column returned from the SQL statement. Must be within the p_min_columns through p_max_columns range.
p_search_column_no	Number of the column used to restrict the SQL statement. Must be within the p_min_columns through p_max_columns range.
p_search_value_list	Array of values to look up.
p_display_extra	If set to TRUE, and a value is not found, the search value is added to the result instead.
p_escape_display_extra	If TRUE, p_search_string is escaped if added as "Display Extra" value.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
apex_application_global_vc_arr2	List of VARCHAR2 indexed by pls_integer. For each entry in p_search_value_list the resulting array contains the value of the first record of the column specified by p_display_column_no in the same order as in p_search_value_list. If no record is found it contains the value of p_search_string if the parameter p_display_extra is set to TRUE. Otherwise the value is skipped.

Example

Looks up the values 7863, 7911 and 7988 and generates a HTML list with the value of the corresponding display column in the LOV query.

```

function render_list (
    p_plugin          in apex_plugin.t_plugin,
    p_item            in apex_plugin.t_page_item,
    p_value           in varchar2,
    p_is_READONLY     in boolean,
    p_is_printer_friendly in boolean )
    return apex_plugin.t_page_item_render_result
is
    l_search_list apex_application_global.vc_arr2;
    l_result_list apex_application_global.vc_arr2;
begin
    l_search_list(1) := '7863';
    l_search_list(2) := '7911';
    l_search_list(3) := '7988';
    --
    l_result_list :=
        apex_plugin_util.get_display_data (
            p_sql_statement      => p_item.lov_definition,
            p_min_columns        => 2,
            p_max_columns        => 2,
            p_component_name     => p_item.name,
            p_search_column_no   => 1,
            p_search_value_list  => l_search_list );
    --
    sys.htp.p('<ul>');
    for i in 1 .. l_result_list.count
    loop
        sys.htp.p(
            '<li>' ||
            sys.htf.escape_sc(l_result_list(i)) ||
            '</li>');
    end loop;
    sys.htp.p('</ul>');
end render_list;

```

45.21 GET_ELEMENT_ATTRIBUTES Function

This function returns some of the standard attributes of an HTML element (such as ID, name, required, placeholder, aria-error-attributes, class) which generates an HTML tag (including input, select, or textarea) to get a consistent set of attributes.

Syntax

```

APEX_PLUGIN_UTIL.GET_ELEMENT_ATTRIBUTES (
    p_item            IN apex_plugin.t_item,
    p_name           IN VARCHAR2 DEFAULT NULL,
    p_default_class IN VARCHAR2 DEFAULT NULL,
    p_add_id         IN BOOLEAN DEFAULT TRUE,
    p_add_required  IN BOOLEAN DEFAULT TRUE,

```

```

    p_add_labelledby      IN BOOLEAN  DEFAULT TRUE,
    p_aria_describedby_id IN VARCHAR2 DEFAULT NULL,
    p_add_multi_value     IN BOOLEAN  DEFAULT FALSE )
RETURN VARCHAR2;

```

Parameters

Parameters	Description
p_item	The p_item parameter of your plug-in function.
p_name	The value returned by apex_plugin.get_input_name_for_page_item.
p_default_class	Default CSS class contained in the result string.
p_add_id	If TRUE, then the ID attribute is also contained in the result string.
p_add_required	Set to TRUE to include the required HTML attribute. Internally, it checks whether p_item is marked as required.
p_add_labelled_by	Set to FALSE to render an HTML input element such as 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 (such as those using div or span) which enable focus for screen reading software.

 **Note:**

Inclusion of aria-labelledby requires the item plug-in to have Standard Form Element set to No and that the item's corresponding label template defines a #LABEL_ID# substitution.

p_aria_describedby_id	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 is used to convey additional information to users of assistive technologies when they are focused on the form field.
p_add_multi_value	If TRUE, renders the required attributes for multiple values (multi-value, multi-value-storage, multi-value-separator). Used for items that support the multiple value infrastructure.

Example

This example emits an INPUT tag of type text which uses apex_plugin_util.get_element_attributes to automatically include the most common attributes.

```

sys.hpt.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||'" ' ||
    ' />');

```

45.22 GET_HTML_ATTR Function

This function returns a properly escaped HTML attribute if `p_value` is not null.

Syntax

```
APEX_PLUGIN_UTIL.GET_HTML_ATTR (
    p_name  IN VARCHAR2,
    p_value IN VARCHAR2 )
return VARCHAR2;
```

Parameters

Parameter	Description
<code>p_name</code>	The HTML attribute in lower case.
<code>p_value</code>	The text string that is escaped.

45.23 GET_ORDERBY_NULLS_SUPPORT Function

This function checks whether the current data source is enabled to specify a `NULLS` clause for sorting. While this is always true for local and REST-enabled SQL, some REST APIs may not support it.

Plug-in developers can use this function to determine whether a `NULLS` clause is possible for this data source and show or hide these options in their UI.

You can specify a `NULLS FIRST` or `NULLS LAST` clause if one of the following conditions is **true**:

- You are working against the local database or a REST-enabled SQL Service.
- The REST API disables pagination. You always fetch all rows and sort locally.
- The REST API disables server-side ordering. You must fetch all rows and sort locally.
- The REST API enables pagination, supports server-side ordering, and includes an `ORDER BY NULLS` clause.

Syntax

```
APEX_PLUGIN_UTIL.GET_ORDERBY_NULLS_SUPPORT
    RETURN apex_exec.t_supports_orderby_nulls_as;
```

Returns

This function returns an instance of `APEX_EXEC.T_SUPPORTS_ORDERBY_NULLS_AS` which indicates whether `ORDER BY NULLS` clauses are supported or how the REST API treats `NULLS` when ordering.

Return	Description
<code>wwv_flow_exec_api.c_orderby_null_s_flexible</code>	The data source supports <code>ORDER BY NULLS</code> clauses.

Return	Description
wwv_flow_exec_api.c_orderby_null_s_are_lowest	The data source treats NULLs as the lowest values when sorting.
wwv_flow_exec_api.c_orderby_null_s_are_highest	The data source treats NULLs as the highest values when sorting.
wwv_flow_exec_api.c_orderby_null_s_always_last	The data source always orders NULLs last.
wwv_flow_exec_api.c_orderby_null_s_always_first	The data source always orders NULLs first.

Example

```

DECLARE
    l_supports_orderby_nulls apex_exec.t_supports_orderby_nulls_as;
BEGIN
    l_supports_orderby_nulls := apex_plugin_util.get_orderby_nulls_support;

    IF l_supports_orderby_nulls = wwv_flow_exec_api.c_orderby_nulls_flexible
THEN
    ...
END IF;
END;

```

45.24 GET_PLSQL_EXPR_RESULT_BOOLEAN Function

This function executes a PL/SQL expression and returns a Boolean result. This function also performs the binding of any bind variables in the provided PL/SQL expression. This function is usually used for plug-in attributes of type PL/SQL expression.

Syntax

```

APEX_PLUGIN_UTIL.GET_PLSQL_EXPR_RESULT_BOOLEAN (
    p_plsql_expression    IN VARCHAR2,
    p_auto_bind_items     IN BOOLEAN      DEFAULT TRUE,
    p_bind_list            IN t_bind_list DEFAULT c_empty_bind_list )
RETURN BOOLEAN;

```

Parameters

Parameter	Description
p_plsql_expression_result	A PL/SQL expression that returns a Boolean.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
BOOLEAN	Boolean result value returned by the PL/SQL function block.

Example

This example executes and returns the result of the PL/SQL expression which is specified in `attribute_03` of an item type plug-in attribute of type PL/SQL Expression.

```
l_result := apex_plugin_util.get_plsql_expr_result_boolean (
    p_plsql_expression => p_item.attribute_03 );
```

45.25 GET_PLSQL_EXPR_RESULT_CLOB Function

This function executes a PL/SQL expression and returns a CLOB result. This function also performs the binding of any bind variables in the provided PL/SQL expression. This function is usually used for plug-in attributes of type PL/SQL expression.

Syntax

```
APEX_PLUGIN_UTIL.GET_PLSQL_EXPR_RESULT_CLOB (
    p_plsql_expression IN VARCHAR2,
    p_auto_bind_items   IN BOOLEAN      DEFAULT TRUE,
    p_bind_list         IN t_bind_list DEFAULT c_empty_bind_list )
RETURN CLOB;
```

Parameters

Parameter	Description
<code>p_plsql_expression</code>	A PL/SQL expression that returns a CLOB.
<code>p_auto_bind_items</code>	Whether to auto-bind APEX items (page and application items).
<code>p_bind_list</code>	Additional bind variables to be used for the SQL query.

Table 45-1 Returns

Return	Description
CLOB	CLOB result value returned by the PL/SQL function block.

Example

```
l_clob := apex_plugin_util.get_plsql_expr_result_clob (
    p_plsql_expression => p_item.attribute_03 );
```

45.26 GET_PLSQL_EXPRESSION_RESULT Function

This function executes a PL/SQL expression and returns a result. This function also performs the binding of any bind variables in the provided PL/SQL expression. This function is usually used for plug-in attributes of type PL/SQL expression.

Syntax

```
APEX_PLUGIN_UTIL.GET_PLSQL_EXPRESSION_RESULT (
    p_plsql_expression IN VARCHAR2,
    p_auto_bind_items  IN BOOLEAN      DEFAULT TRUE,
    p_bind_list        IN t_bind_list DEFAULT c_empty_bind_list )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_plsql_expression_result	A PL/SQL expression that returns a string.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
VARCHAR2	String result value returned by the PL/SQL Expression.

Example

This example executes and returns the result of the PL/SQL expression which is specified in `attribute_03` of an item type plug-in attribute of type PL/SQL Expression.

```
l_result := apex_plugin_util.get_plsql_expression_result (
    p_plsql_expression => p_item.attribute_03 );
```

45.27 GET_PLSQL_FUNC_RESULT_BOOLEAN Function

This function executes a PL/SQL function block and returns the Boolean 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_FUNC_RESULT_BOOLEAN (
    p_plsql_function   IN VARCHAR2,
    p_auto_bind_items  IN BOOLEAN      DEFAULT TRUE,
    p_bind_list        IN t_bind_list DEFAULT c_empty_bind_list )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_plsql_function	A PL/SQL function block that returns a result of type Boolean.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
BOOLEAN	Boolean result value returned by the PL/SQL function block.

Example

The following example executes and returns the Boolean 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_func_result_boolean (
    p_plsql_function => p_item.attribute_03 );
```

45.28 GET_PLSQL_FUNC_RESULT_CLOB Function

This function executes a PL/SQL function block and returns the CLOB result. This function also performs the 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_FUNC_RESULT_CLOB (
    p_plsql_function  IN VARCHAR2,
    p_auto_bind_items IN BOOLEAN      DEFAULT TRUE,
    p_bind_list        IN t_bind_list DEFAULT c_empty_bind_list )
RETURN CLOB;
```

Parameters

Parameter	Description
p_plsql_function	A PL/SQL function block that returns a result of type CLOB.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Table 45-2 Returns

Return	Description
CLOB	CLOB result value returned by the PL/SQL function block.

Example

```
l_clob := apex_plugin_util.get_plsql_func_result_clob (
    p_plsql_function => p_item.attribute_03 );
```

45.29 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,
    p_auto_bind_items   IN BOOLEAN      DEFAULT TRUE,
    p_bind_list          IN t_bind_list DEFAULT c_empty_bind_list )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_plsql_function	A PL/SQL function block that returns a result of type string.
p_auto_bind_items	Whether to auto-bind APEX items (page and application items).
p_bind_list	Additional bind variables to be used for the SQL query.

Return

Return	Description
VARCHAR2	String result value returned by the PL/SQL function block.

Example

The following example executes and returns the result of the PL/SQL function body that is specified in attribute_03 of an item type plug-in attribute of type PL/SQL Function Body.

```
l_result := apex_plugin_util.get_plsql_function_result (
    p_plsql_function => p_item.attribute_03 );
```

45.30 GET_POSITION_IN_LIST Function (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

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

Parameter	Description
p_list	Array of type apex_application_global.vc_arr2 that contains entries of type VARCHAR2.
p_value	Value located in the p_list array.

Return

Return	Description
NUMBER	Returns the position of p_value in the array p_list. If it is not found NULL is returned.

Example

The following example searches for "New York" in the provided list and returns 2 into l_position.

```
DECLARE
    l_list      apex_application_global.vc_arr2;
    l_position  number;
BEGIN
    l_list(1) := 'Rome';
    l_list(2) := 'New York';
    l_list(3) := 'Vienna';

    l_position := apex_plugin_util.get_position_in_list (
                    p_list => l_list,
                    p_value => 'New York' );
END;
```

45.31 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

Parameter	Description
p_search_type	Type of search when used with get_data and get_data2. Use one of the c_search_* constants.
p_search_string	Search string used for the search with get_data and get_data2.

Return

Return	Description
VARCHAR2	Returns p_search_string unchanged. Returns in uppercase if p_search_type is of type c_search_contains_ignore or c_search_exact_ignore.

Example

This example uses a call to get_data or get_data2 to make sure the search string is using the correct case.

```
l_column_value_list :=
    apex_plugin_util.get_data (
        p_sql_statement    => p_item.lov_definition,
        p_min_columns      => 2,
        p_max_columns      => 2,
        p_component_name   => p_item.name,
        p_search_type       => apex_plugin_util.c_search_contains_ignore,
        p_search_column_no  => 1,
        p_search_string     => apex_plugin_util.get_search_string (
            p_search_type   => apex_plugin_util.c_search_contains_ignore,
            p_search_string => p_value ) );
```

45.32 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

```
APEX_PLUGIN_UTIL.GET_VALUE_AS_VARCHAR2 (
    p_data_type      IN VARCHAR2,
    p_value          IN T_VALUE,
    p_format_mask    IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_data_type	The data type of the value stored in p_value.

Parameter	Description
p_value	The value of type t_value which contains the value to be converted and returned as VARCHAR2.
p_format_mask	The format mask used to convert the value into a VARCHAR2.

Example

The following example emits all values stored in the data type aware l_column_value_list array as VARCHAR2.

```

declare
    l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    -- Populate l_column_value_list by calling apex_plugin_util.get_data2
    ...
    -- Emit returned data
    sys.htp.p( '<table>' );
    for l_row in 1 .. l_column_value_list( 1 ).value_list.count
    loop
        sys.htp.p( '<tr>' );
        for l_column in 1 .. l_column_value_list.count loop
            sys.htp.p(
                '<td>' ||
                apex_plugin_util.get_value_as_varchar2 (
                    p_data_type => l_column_value_list( l_column ).data_type,
                    p_value =>
                l_column_value_list( l_column ).value_list( l_row )
                ) ||
                '</td>' );
            end loop;
            sys.htp.p( '</tr>' );
        end loop;
        sys.htp.p( '</table>' );
    end;
end;
```

45.33 GET_WEB_SOURCE_OPERATION Function

This function gets a REST Data Source operation. The REST Data Source operation object contains all meta data for the HTTP request which needs to be done to implement the given database operation (such as INSERT, UPDATE, DELETE).

Syntax

```

APEX_PLUGIN_UTIL.GET_WEB_SOURCE_OPERATION (
    p_web_source      in apex_plugin.t_web_source,
    p_db_operation    in apex_plugin.t_db_operation    DEFAULT NULL,
    p_perform_init    in BOOLEAN                      DEFAULT FALSE,
    p_preserve_headers in BOOLEAN                     DEFAULT FALSE )
RETURN apex_plugin.t_web_source_operation;
```

Parameters

Parameter	Description
p_web_source	REST Data Source plug-in meta data.
p_db_operation	Database operation to look up the Web Source operation (such as UPDATE -> PUT, INSERT -> POST).
p_perform_init	Whether to initialize the HTTP request environment (HTTP request headers, cookies, request body placeholder replacements). If FALSE, the Plug-In developer is responsible for setting up the environment themselves.
p_preserve_headers	Whether to preserve HTTP request headers in apex_web_service.g_request_headers.

Returns

Parameter	Description
*	Plug-In meta data for the web source operation.

Example

The following example uses `get_web_source_operation` as part of a Plug-In "fetch" procedure in order to get meta data about the REST Data Source operation.

```
procedure plugin_fetch(
    p_plugin      in          apex_plugin.t_plugin,
    p_web_source  in          apex_plugin.t_web_source,
    p_params      in          apex_plugin.t_web_source_fetch_params,
    p_result      in out nocopy apex_plugin.t_web_source_fetch_result )
IS
    l_web_source_operation apex_plugin.t_web_source_operation;
BEGIN
    l_web_source_operation := apex_plugin_util.get_web_source_operation(
        p_web_source  => p_web_source,
        p_db_operation => apex_plugin.c_db_operation_fetch_rows,
        p_perform_init => true );
    p_result.responses.extend( 1 );
    apex_plugin_util.make_rest_request(
        p_web_source_operation => l_web_source_operation,
        --
        p_response           => p_result.responses( 1 ),
        p_response_parameters => p_result.out_parameters );
END plugin_fetch;
```

45.34 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

Parameter	Description
p_value1	First value to compare.
p_value2	Second value to compare.

Returns

Return	Description
BOOLEAN	Returns TRUE if both values are equal or both values are NULL, otherwise it returns FALSE.

Example

In the following example, if the value in the database is different from what is entered, the code in the if statement is executed.

```
if NOT apex_plugin_util.is_equal(l_database_value, l_current_value) then
    -- value has changed, do something
    null;
end if;
```

45.35 IS_COMPONENT_USED Function

This function returns TRUE if the passed build option, authorization, and condition are valid to display, process, or use this component.

Syntax

```
APEX_PLUGIN_UTIL.IS_COMPONENT_USED (
    p_build_option_id      IN NUMBER      DEFAULT NULL,
    p_authorization_scheme_id IN VARCHAR2,
    p_condition_type        IN VARCHAR2,
    p_condition_expression1 IN VARCHAR2,
    p_condition_expression2 IN VARCHAR2,
    p_component             IN VARCHAR2 DEFAULT NULL )
RETURN BOOLEAN;
```

45.36 MAKE_REST_REQUEST Procedure Signature 1

This procedure performs the actual REST request (HTTP). Unlike a direct invocation of APEX_WEB_SERVICE.MAKE_REST_REQUEST, this procedure respects all REST Data Source parameters.

Syntax

```
APEX_PLUGIN_UTIL.MAKE_REST_REQUEST (
    p_web_source_operation IN          apex_plugin.t_web_source_operation,
    p_request_body          IN          CLOB      DEFAULT NULL,
    p_bypass_cache          IN          BOOLEAN   DEFAULT FALSE,
    --
    p_time_budget           IN OUT    NOCOPY NUMBER,
    --
    p_response               IN OUT    NOCOPY CLOB,
    p_response_parameters   IN OUT    NOCOPY
apex_plugin.t_web_source_parameters );
```

Parameters

Parameter	Description
p_web_source_operation	Plug-In meta data for the REST Data Source operation.
p_request_body	Override request body to use.
p_bypass_cache	If "true" then the cache is not used.
p_time_budget	If "all rows" are fetched (multiple HTTP requests), then the process stops when the time budget is exhausted and an error raises.

Returns

Parameter	Description
p_time_budget	Time budget left after request has been made.
p_response	Received response of the HTTP invocation.
p_response_parameters	Received response headers and cookies, based on REST Data Source meta data.

Example

The following example demonstrates a simplified Plug-In "fetch" procedure doing HTTP requests with APEX_PLUGIN_UTIL.MAKE_REST_REQUEST.

```
procedure plugin_fetch(
    p_plugin      in      apex_plugin.t_plugin,
    p_web_source in      apex_plugin.t_web_source,
    p_params      in      apex_plugin.t_web_source_fetch_params,
    p_result      in out nocopy apex_plugin.t_web_source_fetch_result )
IS
    l_web_source_operation apex_plugin.t_web_source_operation;
    l_time_budget          pls_integer := 60;
    l_page_to_fetch        pls_integer := 1;
```

```

    l_continue_fetching      boolean;
BEGIN

    l_web_source_operation := apex_plugin_util.get_web_source_operation(
        p_web_source    => p_web_source,
        p_db_operation => apex_plugin.c_db_operation_fetch_rows,
        p_perform_init => true );

    --
    -- loop to execute HTTP request as long as we receive a response header
    named "moreRows"
    -- with the value of "true". A time budget of (initially 60) seconds is
    passed as
    -- IN OUT parameter to MAKE_REST_REQUEST; once that budget is exhausted,
    an error will
    -- be raised.
    --
    while l_continue_fetching loop
        p_result.responses.extend( 1 );
        l_page_to_fetch := l_page_to_fetch + 1;

        apex_plugin_util.make_rest_request(
            p_web_source_operation => l_web_source_operation,
            p_bypass_cache         => false,
            p_time_budget          => l_time_budget,
            --
            p_response             => p_result.responses( l_page_to_fetch ),
            p_response_parameters => p_result.out_parameters );

        l_continue_fetching := false;
        for h in 1 .. apex_web_service.g_headers.count loop
            IF apex_web_service.g_headers( h ).name = 'moreRows' and
                apex_web_service.g_headers( h ).value = 'true'
            THEN
                l_continue_fetching := true;
                exit;
            END IF;
        END LOOP;
    END LOOP;
END plugin_fetch;

```

45.37 MAKE_REST_REQUEST Procedure Signature 2

This procedure performs the actual REST request (HTTP). It uses `apex_web_service`. All parameters for `apex_web_service.make_rest_request` are derived from the REST Data Source meta data passed in as `p_web_source_operation`.

Syntax

```

APEX_PLUGIN_UTIL.MAKE_REST_REQUEST (
    p_web_source_operation IN              apex_plugin.t_web_source_operation,
    --
    p_request_body       IN              CLOB DEFAULT NULL,
    --

```

```

    p_response          IN OUT NOCOPY CLOB,
    p_response_parameters IN OUT NOCOPY
apex_plugin.t_web_source_parameters );

```

Parameters

Parameter	Description
p_web_source_operation	Plug-in meta data for the REST Data Source operation.
p_bypass_cache	If TRUE, then the cache is not used.
p_request_body	Override request body to use.

Returns

Parameter	Description
p_response	Received response of the HTTP invocation.
p_response_parameters	Received response headers and cookies, based on REST Data Source meta data.

Example

The following example demonstrates a simplified Plug-In "fetch" procedure doing a HTTP request with APEX_PLUGIN_UTIL.MAKE_REST_REQUEST.

```

procedure plugin_fetch(
    p_plugin      in      apex_plugin.t_plugin,
    p_web_source  in      apex_plugin.t_web_source,
    p_params       in      apex_plugin.t_web_source_fetch_params,
    p_result       in out nocopy apex_plugin.t_web_source_fetch_result )
is
    l_web_source_operation apex_plugin.t_web_source_operation;
BEGIN
    l_web_source_operation := apex_plugin_util.get_web_source_operation(
        p_web_source  => p_web_source,
        p_db_operation => apex_plugin.c_db_operation_fetch_rows,
        p_perform_init => true );

    p_result.responses.extend( 1 );

    apex_plugin_util.make_rest_request(
        p_web_source_operation => l_web_source_operation,
        --
        p_response           => p_result.responses( 1 ),
        p_response_parameters => p_result.out_parameters );
END plugin_fetch;

```

45.38 PAGE_ITEM_NAMES_TO_JQUERY Function (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

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

Parameter	Description
p_page_item_names	Comma-delimited list of page item names.

Return

Return	Description
VARCHAR2	Transforms the page items specified in p_page_item_names into a jQuery selector.

Example

The following example shows the code to construct the initialization call for a JavaScript function called `myOwnWidget`. This function gets an object with several attributes where one attribute is `pageItemsToSubmit` which is expected to be a jQuery selector.

```
apex_javascript.add_onload_code (
    p_code => 'myOwnWidget(' ||
        '"#' || p_item.name || ',', ' ||
        '{' ||
        apex_javascript.add_attribute('ajaxIdentifier',
apex_plugin.get_ajax_identifier) ||
            apex_javascript.add_attribute('dependingOnSelector',
apex_plugin_util.page_item_names_to_jquery(p_item.lov_cascade_parent_items)) ||
                apex_javascript.add_attribute('optimizeRefresh',
p_item.ajax_optimize_refresh) ||
                    apex_javascript.add_attribute('pageItemsToSubmit',
apex_plugin_util.page_item_names_to_jquery(p_item.ajax_items_to_submit)) ||
                        apex_javascript.add_attribute('nullValue',
```

```
p_item.lov_null_value, false, false)||  
    '});' );
```

45.39 PARSE_REFETCH_RESPONSE Function

This function parses the response from a "DML row refetch." A "row refetch" is used for lost update detection in order to verify that nobody else changed the row.

To use this function, the REST Data Source must have a "Fetch Single Row" database operation defined.

This procedure always parses JSON at the root level; an array column cannot be passed in. Array column contents can be accessed in the values context using [OPEN_ARRAY Procedure Signature 1](#) and [CLOSE_ARRAY Procedure](#) in APEX_EXEC.

Syntax

```
APEX_PLUGIN_UTIL.PARSE_REFETCH_RESPONSE (  
    p_web_source_operation IN apex_plugin.t_web_source_operation,  
    p_web_source          IN apex_plugin.t_web_source,  
    p_values_context       IN apex_exec.t_context,  
    --  
    p_response            IN CLOB )  
RETURN apex_exec.t_context;
```

Parameters

Parameter	Description
p_web_source_operation	REST Data Source operation (Plug-In) meta data.
p_web_source	REST Data Source (Plug-In) meta data.
p_response	REST response to parse.
p_values_context	Values context, needed for DML column definitions.

Returns

Parameter	Description
*	APEX_EXEC "Values" context object for the plug-in developer to retrieve the checksum or column values.

Example

The following example demonstrates how to perform a "refetch" operation within the Plug-In DML function for a given row to be updated and compare checksums in order to detect lost updates.

```
procedure plugin_dml(  
    p_plugin      in      apex_plugin.t_plugin,  
    p_web_source in      apex_plugin.t_web_source,  
    p_params      in      apex_plugin.t_web_source_dml_params,  
    p_result      in out nocopy apex_plugin.t_web_source_dml_result )  
IS  
    l_web_source_operation apex_plugin.t_web_source_operation;
```

```

l_request_body      clob;
l_response         clob;

l_refetch_context   apex_exec.t_context;
l_checksum          varchar2(32767);
l_refetched_checksum varchar2(32767);

BEGIN
  p_result.update_values_context := p_params.update_values_context;

  --
  -- this code performs a "refetch" operation for a row, in order to perform
  -- lost update detection. This happens before the actual DML.
  --
  IF p_web_source.operations.exists( apex_plugin.c_db_operation_fetch_row )
THEN

  l_web_source_operation := apex_plugin_util.get_web_source_operation(
    p_web_source      => p_web_source,
    p_db_operation    => apex_plugin.c_db_operation_fetch_row,
    p_preserve_headers => false,
    p_perform_init     => true );

  -- add some logic to add primary key values to the URL or as HTTP headers
here
  -- PK values can be obtained from "p_params.update_values_context"

  apex_plugin_util.make_rest_request(
    p_web_source_operation => l_web_source_operation,
    p_request_body        => l_request_body,
    p_response            => l_response,
    p_response_parameters => p_result.out_parameters );

  l_refetch_context := apex_plugin_util.parse_refetch_response(
    p_web_source_operation => l_web_source_operation,
    p_web_source           => p_web_source,
    p_response              => l_response,
    p_values_context        => p_params.update_values_context );

  IF apex_exec.next_row( p_context => l_refetch_context ) THEN

    l_checksum           := apex_exec.get_row_version_checksum( p_context
=> p_params.update_values_context );
    l_refetched_checksum := apex_exec.get_row_version_checksum( p_context
=> l_refetch_context );

    IF l_checksum != l_refetched_checksum THEN
      apex_exec.set_row_status(
        p_context  => p_result.update_values_context,
        p_sqlcode   => -20987,
        p_sqlerrm   => 'APEX.DATA_HAS_CHANGED' );
    END IF;
    END IF;
  END IF;

  -- continue with DML logic here ...

```

```
END plugin_dml;
```

 **See Also:**

- [CLOSE_ARRAY Procedure](#)
- [OPEN_ARRAY Procedure Signature 1](#)

45.40 PRINT_DISPLAY_ONLY Procedure Signature 1 (Deprecated)

 **Caution:**

This API is deprecated and will be removed in a future release.

Use [PRINT_READ_ONLY Procedure Signature 1](#) instead.

This procedure outputs a SPAN tag for a display-only field.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_DISPLAY_ONLY (
    p_item_name      IN VARCHAR2,
    p_display_value  IN VARCHAR2,
    p_show_line_breaks IN BOOLEAN,
    p_attributes     IN VARCHAR2,
    p_id_postfix     IN VARCHAR2 DEFAULT '_DISPLAY' );
```

Parameters

Parameter	Description
p_item_name	Name of the page item. This parameter should be called with <code>p_item.name</code> .
p_display_value	Text to be displayed.
p_show_line_breaks	If set to TRUE line breaks in <code>p_display_value</code> are changed to <code>
</code> so that the browser renders them as line breaks.
p_attributes	Additional attributes added to the SPAN tag.
p_id_postfix	Postfix which is getting added to the value in <code>p_item_name</code> to get the ID for the SPAN tag. Default is <code>_DISPLAY</code> .

Example

The following example could be used in an item type plug-in to render a display-only page item.

```
apex_plugin_util.print_display_only (
    p_item_name      => p_item.name,
    p_display_value   => p_value,
    p_show_line_breaks => false,
    p_escape          => true,
    p_attributes       => p_item.element_attributes );
```

45.41 PRINT_DISPLAY_ONLY Procedure Signature 2 (Deprecated)

Caution:

This API is deprecated and will be removed in a future release.

Use [PRINT_READ_ONLY Procedure Signature 2](#) instead.

This procedure outputs a SPAN tag for a display-only field.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_DISPLAY_ONLY (
    p_item           IN apex_plugin_util.t_item,
    p_display_value  IN apex_session_state.t_value,
    p_show_line_breaks IN BOOLEAN,
    p_escape          IN BOOLEAN  DEFAULT NULL,
    p_id_postfix     IN VARCHAR2 DEFAULT '_DISPLAY',
    p_show_icon       IN BOOLEAN  DEFAULT TRUE );
```

Parameters

Parameter	Description
p_item	The p_item record to be passed in.
p_display_value	Text to be displayed. p_param.session_state_value should be passed in.
p_show_line_breaks	If set to TRUE line breaks in p_display_value are changed to so that the browser renders them as line breaks.
p_escape	Whether to escape the value. If p_escape is unspecified, the value from p_item is used.
p_id_postfix	Postfix which is getting added to the value in p_item.name to get the ID for the SPAN tag. Default is _DISPLAY.
p_show_icon	Whether to render the item icon. Default is TRUE.

Example

The following example could be used in an item type plug-in to render a display-only page item.

```
apex_plugin_util.print_display_only (
    p_item      => p_item,
    p_display_value => p_param.session_state_value );
```

45.42 PRINT_ESCAPED_VALUE Procedure Signature 1

This procedure outputs the value in an escaped form and chunks big strings into smaller outputs.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_ESCAPED_VALUE (
    p_value    IN VARCHAR2 );
```

Parameters

Parameter	Description
p_value	Text which should be escaped and then printed to the HTTP buffer.

Example

Prints a hidden field with the current value of the page item.

```
sys.http.prn('<input type="hidden" name="'||l_name||'" id="'||p_item_name||"
value="');
apex_plugin_util.print_escaped_value( p_value );
sys.http.prn('>');
```

45.43 PRINT_ESCAPED_VALUE Procedure Signature 2

This procedure outputs the value in an escaped form and chunks big strings into smaller outputs.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_ESCAPED_VALUE (
    p_value IN apex_session_state.t_value );
```

Parameters

Parameter	Description
p_value	Text which should be escaped and then printed to the HTTP buffer.

Example

Prints a hidden field with the current value of the page item.

```
sys.htp.prn('<input type="hidden" name="'||p_item.name||'" id="'||  
p_item.name||'" value="');  
apex_plugin_util.print_escaped_value( p_param.session_state_value );  
sys.htp.prn('>');
```

45.44 PRINT_HIDDEN Procedure

This procedure outputs a hidden field to store the page item value.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_HIDDEN (  
    p_item      IN apex_plugin.t_item,  
    p_param     IN apex_plugin.t_item_render_param,  
    p_id_postfix IN VARCHAR2 DEFAULT NULL,  
    p_classes   IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_item	The p_item record to be passed in.
p_param	The p_param record to be passed in.
p_id_postfix	A postfix for the ID of the hidden element. It is appended to the item's name.
p_classes	Additional classes for the hidden element.

Example

The following example renders a hidden element in an item type plug-in.

```
apex_plugin_util.print_hidden (  
    p_item  => p_item,  
    p_param => p_param );
```

45.45 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

Parameter	Description
p_item_name	Name of the page item. For this parameter the <code>p_item.name</code> should be passed in.
p_value	Current value of the page item. For this parameter <code>p_value</code> should be passed in.
p_is_READONLY	Is the item rendered readonly. For this parameter <code>p_is_READONLY</code> should be passed in.
p_is_printer_friendly	Is the item rendered in printer friendly mode. For this parameter <code>p_is_printer_friendly</code> should be passed in.
p_id_postfix	Used to generate the ID attribute of the hidden field. It is build based on <code>p_item_name</code> and the value in <code>p_id_postfix</code> .

Example

Writes a hidden field with the current value to the HTTP output if `p_is_READONLY` is TRUE and `p_printer_friendly` is FALSE.

```
apex_plugin_util.print_hidden_if_READONLY (
    p_item_name      => p_item.name,
    p_value          => p_value,
    p_is_READONLY    => p_is_READONLY,
    p_is_printer_friendly => p_is_printer_friendly );
```

45.46 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.hsf.escape_sc(l_values(i).display_value), false, true) ||
    apex_javascript.add_attribute('r',
sys.hsf.escape_sc(l_values(i).return_value), false, false) ||
    '}');
end loop;
-- close the JSON structure
sys.htp.p(']');
```

45.47 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

Parameter	Description
p_sql_statement	A SQL statement which returns two columns from the SELECT.
p_component_name	The name of the page item or report column that is used in case an error is displayed.
p_escape	If set to TRUE the value of the display column is escaped, otherwise it is output as is.
p_replace_substitutions	If set to TRUE, apex_plugin_util.replace_substitutions is called for the value of the display column, otherwise, it is output as is.

Example

This example shows how to use the procedure in an Ajax callback function of an item type plug-in. The following call writes the LOV result as a JSON array to the HTTP output.

```
apex_plugin_util.print_lov_as_json (
    p_sql_statement => p_item.lov_definition,
```

```
p_component_name => p_item.name,
p_escape          => true );
```

45.48 PRINT_OPTION Procedure

This procedure outputs an OPTION tag.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_OPTION (
    p_display_value      IN VARCHAR2,
    p_return_value       IN VARCHAR2,
    p_is_selected        IN BOOLEAN,
    p_attributes         IN VARCHAR2,
    p_escape              IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
p_display_value	Text which is displayed by the option.
p_return_value	Value which is set when the option is picked.
p_is_selected	Set to TRUE if the selected attribute should be set for this option.
p_attributes	Additional HTML attributes which should be set for the OPTION tag.
p_escape	Set to TRUE if special characters in p_display_value should be escaped.

Example

The following example could be used in an item type plug-in to create a SELECT list. Use apex_plugin_util.is_equal to find out which list entry should be marked as current.

```
sys.htp.p('<select id="'||p_item.name||'" size=""||nvl(p_item.element_height,
5)||'"'||coalesce(p_item.element_attributes,
'class="new_select_list"')||'>');
-- loop through the result and add list entries
for i in 1 .. l_values.count
loop
    apex_plugin_util.print_option (
        p_display_value => l_values(i).display_value,
        p_return_value  => l_values(i).return_value,
        p_is_selected   =>
apex_plugin_util.is_equal(l_values(i).return_value, p_value),
        p_attributes    => p_item.element_option_attributes,
        p_escape         => true );
end loop;
sys.htp.p('</select>');
```

45.49 PRINT_READ_ONLY Procedure Signature 1

This procedure outputs a read-only text field or textarea. Use when displaying a single value.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_READ_ONLY (
    p_item              IN apex_plugin_api.t_item,
    p_param             IN apex_plugin_api.t_item_render_param,
    p_value             IN apex_session_state_api.t_value
                           DEFAULT
    apex_session_state_api.t_value(),
    p_display_value     IN VARCHAR2          DEFAULT
    c_ignore_display_value,
    p_width             IN PLS_INTEGER      DEFAULT
    NULL,
    p_height            IN PLS_INTEGER      DEFAULT
    NULL,
    p_css_classes       IN VARCHAR2         DEFAULT
    NULL,
    p_protected         IN BOOLEAN          DEFAULT
    TRUE,
    p_escape             IN BOOLEAN          DEFAULT
    TRUE )
```

Parameters

Parameter	Description
p_item	The item's p_item variable.
p_param	The item's p_param variable.
p_value	(Optional) The unescaped value (API always escapes it). If not passed, defaults to p_param.session_state_value.
p_display_value	(Optional) Used as the display value. If not passed, defaults to c_ignore_display_value and is ignored.
p_width	(Optional) The width of the item. If not passed, uses p_item.element_width.
p_height	(Optional) The height of the item. If not passed, uses p_item.element_height. If height is greater than 1, API renders a textarea instead of a text field.
p_css_classes	(Optional) Additional CSS classes to be added to the text field or textarea.
p_protected	Add checksum for the value. Default TRUE.
p_escape	Controls escaping of p_display_value (p_value is always escaped). Default TRUE.

Example

```
apex_plugin_util.print_read_only (
    p_item      => p_item,
    p_param     => p_param,
```

```
p_display_value => l_display_value,
p_css_classes   => 'my_READONLY-custom-item' );
```

45.50 PRINT_READ_ONLY Procedure Signature 2

This procedure outputs a read-only text field or textarea. Use when displaying multiple values.

Syntax

```
APEX_PLUGIN_UTIL.PRINT_READ_ONLY (
    p_item          IN apex_plugin_api.t_item,
    p_param         IN apex_plugin_api.t_item_render_param,
    p_value         IN apex_session_state_api.t_value
                           DEFAULT apex_session_state_api.t_value(),
    p_display_values IN apex_application_global.vc_arr2,
    p_width         IN PLS_INTEGER
                           DEFAULT NULL,
    p_height        IN PLS_INTEGER
                           DEFAULT NULL,
    p_css_classes   IN VARCHAR2
                           DEFAULT NULL,
    p_protected     IN BOOLEAN
                           DEFAULT TRUE,
    p_escape         IN BOOLEAN
                           DEFAULT TRUE )
```

Parameters

Parameter	Description
p_item	The item's p_item variable.
p_param	The item's p_param variable.
p_value	(Optional) The unescaped value (API always escapes it). If NULL, defaults to p_param.session_state_value.
p_display_values	Array of display values.
p_width	(Optional) The width of the item. If NULL, uses p_item.element_width.
p_height	(Optional) The height of the item. If NULL, uses p_item.element_height. If height is greater than 1, API renders a textarea instead of a text field.
p_css_classes	(Optional) Additional CSS classes to be added to the text field or textarea.
p_protected	Add checksum for the value. Default TRUE.
p_escape	Controls escaping of p_display_values (p_value is always escaped). Default TRUE.

Example

```
procedure render_custom_item (
    p_item      in      apex_plugin.t_item,
    p_plugin    in      apex_plugin.t_plugin,
    p_param     in      apex_plugin.t_item_render_param,
    p_result    in out nocopy apex_plugin.t_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_col_no      => 1,
    p_search_value_list   => l_search_list );
apex_plugin_util.print_read_only (
    p_item               => p_item,
    p_param              => p_param,
    p_display_values     => l_result_list,
    p_css_classes        => 'my_READONLY_CUSTOM_ITEM' );
END render_custom_item;

```

45.51 PROCESS_DML_RESPONSE Procedure

This procedure parses the DML request response and load return values to the values context object.

This procedure always parses JSON at the root level; an array column cannot be passed in. Array column contents can be accessed in the values context using [OPEN_ARRAY Procedure Signature 1](#) and [CLOSE_ARRAY Procedure](#) in APEX_EXEC.

Syntax

```
APEX_PLUGIN_UTIL.PROCESS_DML_RESPONSE (
    p_web_source_operation IN apex_plugin.t_web_source_operation,
    p_web_source          IN apex_plugin.t_web_source,
    --
    p_response            IN CLOB,
    p_status_code         IN pls_integer,
    p_error_message       IN VARCHAR2,
    --
    p_values_context      IN apex_exec.t_context );
```

Parameters

Parameter	Description
p_web_source_operation	REST Data Source operation (Plug-In) meta data.
p_web_source	REST Data Source (Plug-In) meta data.
p_response	REST response to parse.
p_status_code	HTTP status code to use.
p_error_message	Error message to use.
p_values_context	Values context to store the return values in.

Example

The following example uses PROCESS_DML_RESPONSE within a plug-in DML procedure.

```
procedure plugin_dml(
    p_plugin      in          apex_plugin.t_plugin,
    p_web_source  in          apex_plugin.t_web_source,
    p_params      in          apex_plugin.t_web_source_dml_params,
    p_result      in out nocopy apex_plugin.t_web_source_dml_result )
IS
    l_web_source_operation apex_plugin.t_web_source_operation;
    l_request_body         clob;
    l_response             clob;
    l_return_values_ctx    apex_exec.t_context := 
p_params.insert_values_context;
BEGIN
    l_web_source_operation := apex_plugin_util.get_web_source_operation(
        p_web_source  => p_web_source,
        p_db_operation => apex_plugin.c_db_operation_insert,
        p_perform_init => true );
    apex_plugin_util.build_request_body(
        p_request_format      => apex_plugin.c_format_json,
        p_profile_columns     => p_web_source.profile_columns,
        p_values_context       => p_params.insert_values_context,
        p_build_when_empty    => true,
        p_request_body         => l_request_body );
    -- continue with APEX_PLUGIN_UTIL.MAKE_REST_REQUEST
    apex_plugin_util.process_dml_response(
        p_web_source_operation => l_web_source_operation,
        p_web_source           => p_web_source,
        --
        p_response              => l_response,
        --
        p_status_code           => apex_web_service.g_status_code,
        p_error_message          => apex_web_service.g_reason_phrase,
        --
        p_values_context         => l_return_values_ctx );
END plugin_dml;
```

 **See Also:**

- [CLOSE_ARRAY Procedure](#)
- [OPEN_ARRAY Procedure Signature 1](#)

45.52 REPLACE_SUBSTITUTIONS Function

This function replaces any &ITEM. substitution references with their actual value. If p_escape is set to TRUE, any special characters contained in the value of the referenced item are escaped to prevent Cross-site scripting (XSS) attacks.

Syntax

```
APEX_PLUGIN_UTIL.REPLACE_SUBSTITUTIONS (
    p_value      IN VARCHAR2,
    p_escape     IN BOOLEAN  DEFAULT TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_value	This value is a string which can contain several &ITEM references which are replaced by their actual page item values.
p_escape	If set to TRUE any special characters contained in the value of the referenced item are escaped to prevent Cross-site scripting (XSS) attacks. If set to FALSE, the referenced items are not escaped.

Example

The following example replaces any substitution syntax references in the region plug-in attribute 05 with their actual values. Any special characters in the values are escaped.

```
l_advanced_formatting := apex_plugin_util.replace_substitutions (
    p_value => p_region.attribute_05,
    p_escape => true );
```

45.53 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

```
APEX_PLUGIN_UTIL.SET_COMPONENT_VALUES (
    p_column_value_list IN t_column_list,
    p_row_num           IN PLS_INTEGER );
```

Parameters

Parameter	Description
p_column_value_list	Table of t_column_list returned by the call to APEX_PLUGIN.GET_DATA2.

Parameter	Description
p_row_num	Row number in p_column_value_list for which the column values should be set in Session State.

Example

This example is the 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.

```

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 allows a developer to
    -- reference a column of the SQL query
    -- which should be used as the text for the link.
    c_link_label_column constant varchar2(128) := p_item.attribute_02;
    --
    l_column_value_list apex_plugin_util.t_column_value_list2;
begin
    l_column_value_list :=
        apex_plugin_util.get_data2 (
            p_sql_statement =>
            ... );
    --
    sys.htp.p('<ul>');
    for i in 1 .. l_column_value_list.count(1)
    loop
        -- Set all column values of the current row
        apex_plugin_util.set_component_values (
            p_column_value_list => l_column_value_list,
            p_row_num           => i );
        --
        sys.htp.p(
            '<li><a href="' ||

apex_escape.html_attribute( apex_util.prepare_url( c_link_target )) || '">' ||
            apex_escape.html( v( c_link_label_column )) ||

'</a></li>');

        --
        apex_plugin_util.clear_component_values;

```

```
    end loop;
    sys.hpt.p('<ul>');
end;
```

45.54 SPLIT_MUTLIPLE_VALUE_TO_TABLE Function

This function converts a separated input string into an array.

Syntax

```
APEX_PLUGIN_UTIL.SPLIT_MUTLIPLE_VALUE_TO_TABLE (
    p_value IN CLOB,
    p_item  IN apex_plugin_api.t_item )
RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_value	The input value.
p_item	This type contains information about the current item.

Returns

An array of strings.

Example

```
DECLARE
    l_arr  apex_t_varchar2 := apex_t_varchar2();
    l_item apex_plugin.t_item;
BEGIN
    l_arr := apex_plugin_util.split_multiple_value_to_table(
        p_value => '['"dog","cat","capybara"]',
        p_item   => l_item
    );
END;
/
-> apex_t_varchar2('dog','cat','capybara')
```

46

APEX_PRINT

The APEX_PRINT package provides APIs to invoke remote print servers and generate documents based on templates and data.

- Constants
- GENERATE_DOCUMENT Function Signature 1
- GENERATE_DOCUMENT Function Signature 2
- GENERATE_DOCUMENT Function Signature 3
- GENERATE_DOCUMENT Function Signature 4
- REMOVE_TEMPLATE Procedure
- UPLOAD_TEMPLATE Function

46.1 Constants

The APEX_PRINT package uses the following constants.

Template Type Constants

```
subtype t_template_type is pls_integer range 1..10;

c_template_docx      constant t_template_type := 1;
c_template_xlsx     constant t_template_type := 2;
c_template_pptx     constant t_template_type := 3;
c_template_html      constant t_template_type := 4;
c_template_markdown constant t_template_type := 5;
c_template_csv       constant t_template_type := 6;
c_template_txt       constant t_template_type := 7;
c_template_ods       constant t_template_type := 8;
c_template_odt       constant t_template_type := 9;
c_template_odp       constant t_template_type := 10;
```

Output Type Constants

```
subtype t_output_type is pls_integer range 1..11;

c_output_pdf        constant t_output_type := 1;
c_output_docx       constant t_output_type := 2;
c_output_xlsx       constant t_output_type := 3;
c_output_pptx       constant t_output_type := 5;
c_output_html        constant t_output_type := 4;
c_output_markdown   constant t_output_type := 6;
c_output_csv         constant t_output_type := 7;
c_output_txt         constant t_output_type := 8;
c_output_odt         constant t_output_type := 9;
```

```
c_output_ods      constant t_output_type := 10;
c_output_odp      constant t_output_type := 11;
```

46.2 GENERATE_DOCUMENT Function Signature 1

This function generates a document based on data and a template and returns the contents.

Can only be used when Oracle Document Generator Pre-built Function is configured as print server in the instance.

To be used when printing a single document using a custom template, which is not stored as report layout.

Syntax

```
APEX_PRINT.GENERATE_DOCUMENT (
    p_data          IN CLOB,
    p_template      IN BLOB,
    p_template_type IN t_template_type  DEFAULT c_template_docx,
    p_output_type   IN t_output_type    DEFAULT c_output_pdf )
RETURN BLOB;
```

Parameters

Parameter	Description
p_data	Data for the document. Currently JSON format only.
p_template	Contents of the template.
p_template_type	Type of the template.
p_output_type	The type of document.

Returns

A BLOB containing the generated document.

Example

The following example generates a PDF document using an uploaded template and custom JSON data.

```
DECLARE
    l_template blob;
    l_data      sys.json_object_t := sys.json_object_t();
    l_document  blob;
BEGIN

    SELECT blob_content
        INTO l_template
        FROM apex_application_temp_files
       WHERE name = :P1_TEMPLATE;

    l_data.put( 'name', 'Scott' );

    l_document := apex_print.generate_document(
                    p_data           => l_data.to_clob,
```

```

    p_template      => l_template );
END;

```

46.3 GENERATE_DOCUMENT Function Signature 2

This function returns a document as BLOB using a pre-defined report query.

Syntax

```

APEX_PRINT.GENERATE_DOCUMENT (
    p_application_id          IN NUMBER,
    p_report_query_static_id   IN VARCHAR2,
    p_report_layout_static_id  IN VARCHAR2          DEFAULT NULL,
    p_output_type               IN t_output_type    DEFAULT c_output_pdf )
RETURN BLOB;

```

Parameters

Parameter	Description
p_application_id	Defines the application ID of the report layout.
p_report_query_static_id	Static ID of the report query (stored under application's shared components).
p_report_layout_static_id	Static ID of the report layout (stored under application's shared components).
p_output_type	Defines the document format. See <code>t_output_type</code> for the available types in Constants .

Returns

A BLOB containing the generated document.

Example

The following example generates a PDF document using a report query and a report layout defined in an application.

```

DECLARE
    l_document blob;
BEGIN

    l_document :=
        apex_print.generate_document (
            p_application_id      => 100,
            p_report_query_static_id => 'MY_REPORT_QUERY',
            p_report_layout_static_id => 'MY_REPORT_LAYOUT',
            p_output_type           => apex_print.c_output_pdf );

    apex_http.download(
        p_blob      => l_document,
        p_content_type => 'application/pdf',
        p_filename   => 'my-report.pdf' );

```

```
END;
```



See Also:

[Constants](#)

46.4 GENERATE_DOCUMENT Function Signature 3

This function returns a document as BLOB using a pre-defined report layout.

Syntax

```
APEX_PRINT.GENERATE_DOCUMENT (
    p_application_id           IN NUMBER,
    p_data                      IN CLOB,
    p_report_layout_static_id   IN VARCHAR2,
    p_output_type                IN t_output_type      DEFAULT c_output_pdf)
RETURN BLOB;
```

Parameters

Parameter	Description
p_application_id	Defines the application ID of the report layout.
p_data	Report data. The format depends on the type of print server that is used.
p_report_layout_static_id	Static ID of the report layout (stored under application's shared components).
p_output_type	Defines the document format. See <code>t_output_type</code> for the available types in Constants .

Returns

A BLOB containing the generated document.

Example

The following example generates a PDF document using custom JSON and a report layout defined in an application.

```
DECLARE
    l_document blob;
    l_json      sys.json_object_t := sys.json_object_t();
BEGIN
    l_json.put( 'title', 'Hello World' );

    l_document :=
        apex_print.generate_document (
            p_application_id           => 100,
```

```

        p_report_data          => l_json.to_clob(),
        p_report_layout_static_id => 'MY_REPORT_LAYOUT',
        p_output_type           => apex_print.c_output_pdf );

apex_http.download(
    p_blob      => l_document,
    p_content_type => 'application/pdf',
    p_filename   => 'hello-world.pdf' );

END;

```

 See Also:[Constants](#)

46.5 GENERATE_DOCUMENT Function Signature 4

This function generates a document using an uploaded template and returns the contents.

Can only be used when Oracle Document Generator Pre-built Function is configured as print server in the instance.

To be used in combination with the [UPLOAD_TEMPLATE Function](#) and [REMOVE_TEMPLATE Procedure](#) APIs to generate documents using the same custom template, which is not stored as a report layout.

Syntax

```
APEX_PRINT.GENERATE_DOCUMENT (
    p_data      IN CLOB,
    p_template_id IN NUMBER,
    p_output_type  IN t_output_type    DEFAULT c_output_pdf )
RETURN BLOB;
```

Parameters

Parameter	Description
p_data	Data for the document. Currently JSON format only.
p_template_id	ID of the the template.
p_output_type	Static ID of the report layout (stored under application's shared components).
p_output_type	The type of document.

Returns

A BLOB containing the generated document.

Example

See [UPLOAD_TEMPLATE Function](#).

 See Also:

- [UPLOAD_TEMPLATE Function](#)
- [REMOVE_TEMPLATE Procedure](#)

46.6 REMOVE_TEMPLATE Procedure

This procedure removes a template from OCI Object Storage.

Can only be used when Oracle Document Generator Pre-built Function is configured as print server in the instance.

To be used in combination with [UPLOAD_TEMPLATE Function](#) to generate documents using the same custom template, which is not stored as a report layout.

Syntax

```
APEX_PRINT.REMOVE_TEMPLATE (
    p_template_id IN NUMBER )
```

Parameters

Parameter	Description
p_template_id	ID of the the template.

Example

See [UPLOAD_TEMPLATE Function](#).

 See Also:

- [UPLOAD_TEMPLATE Function](#)

46.7 UPLOAD_TEMPLATE Function

This function uploads a template to OCI Object Storage and returns its corresponding ID.

Can only be used when Oracle Document Generator Pre-built Function is configured as print server in the instance.

To be used in combination with the [APEX_PRINT.GENERATE_DOCUMENT](#) and [REMOVE_TEMPLATE Procedure](#) APIs to generate documents using the same custom template, which is not stored as a report layout.

Syntax

```
APEX_PRINT.UPLOAD_TEMPLATE (
    p_template      IN BLOB,
```

```
p_template_type IN t_template_type DEFAULT c_template_docx )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_template	Content of the template.
p_template_type	Type of the template.

Returns

A number containing the unique ID to reference the template in future calls.

Example

The following example uploads the template to Object Storage that was uploaded in Oracle APEX by an end user, generates a PDF document, and removes the template afterwards.

```
DECLARE
    l_template      blob;
    l_template_id   number;
    l_data          sys.json_object_t := sys.json_object_t();
    l_document      blob;
BEGIN

    SELECT blob_content
        INTO l_template
        FROM apex_application_temp_files
       WHERE name = :P1 TEMPLATE;

    l_template_id := apex_print.upload_template( p_template => l_template );

    l_data.put( 'name', 'Scott' );

    l_document := apex_print.generate_document(
                    p_data      => l_data.to_clob,
                    p_template_id => l_template_id );

    apex_print.remove_template( p_template_id => l_template_id );

EXCEPTION
    WHEN others THEN
        apex_print.remove_template( p_template_id => l_template_id );
END;
```

See Also:

- [REMOVE_TEMPLATE Procedure](#)

APEX_PWA

Utilities include: subscribing and unsubscribing users for push notifications; verifying subscription for push notifications; and sending push notifications to subscribed users.

This package is used to provide utilities to applications that have enabled Progressive Web App (PWA).

- [GENERATE_PUSH_CREDENTIALS Procedure](#)
- [HAS_PUSH_SUBSCRIPTION Function](#)
- [PUSH_QUEUE Procedure](#)
- [SEND_PUSH_NOTIFICATION Procedure](#)
- [SUBSCRIBE_PUSH_NOTIFICATIONS Procedure](#)
- [UNSUBSCRIBE_PUSH_NOTIFICATIONS Procedure](#)

47.1 GENERATE_PUSH_CREDENTIALS Procedure

This procedure regenerates push credential keys based on the provided application ID.

Syntax

```
APEX_PWA.GENERATE_PUSH_CREDENTIALS (
    p_application_id IN NUMBER DEFAULT [current application id] )
```

Parameters

Parameter	Description
p_application_id	ID of the application. Defaults to current application.

Example

The following example regenerates push credential keys for application 100.

```
BEGIN
    apex_pwa.generate_push_credentials (
        p_application_id => 100 );
END;
```

47.2 HAS_PUSH_SUBSCRIPTION Function

This function returns whether a user has at least one device subscribed to push notifications.

Syntax

```
APEX_PWA.HAS_PUSH_SUBSCRIPTION (
    p_application_id IN NUMBER    DEFAULT [current application id],
    p_user_name      IN VARCHAR2 DEFAULT [current user] )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_application_id	ID of the application that has the push subscription.
p_user_name	Username of the user that has the push subscription.

Returns

Function returns boolean containing whether a user is subscribed to an application.

Example

The following example verifies whether user "SMITH" has a push subscription for application 100.

```
BEGIN
    apex_pwa.has_push_subscription (
        p_application_id => 100,
        p_user_name      => 'SMITH' );
END;
```

47.3 PUSH_QUEUE Procedure

This procedure triggers the database job to send all push notifications in the queue.

Syntax

```
APEX_PWA.PUSH_QUEUE;
```

Parameters

None.

Example

```
BEGIN
    apex_pwa.push_queue;
END;
```

47.4 SEND_PUSH_NOTIFICATION Procedure

This procedure sends a push notification to a user. All devices that the user subscribes on receive the push notification.

Syntax

```
APEX_PWA.SEND_PUSH_NOTIFICATION (
    p_application_id IN NUMBER      DEFAULT [current application id],
    p_user_name       IN VARCHAR2,
    p_title           IN VARCHAR2,
    p_body             IN VARCHAR2 DEFAULT NULL,
    p_icon_url         IN VARCHAR2 DEFAULT NULL,
    p_target_url       IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_application_id	ID of the application that contains the user to send the push notification to. Defaults to current application.
p_user_name	Username of the user receiving the push notification.
p_title	Title of the push notification.
p_body	Body of the push notification.
p_icon_url	URL of the icon that displays on the push notification. Defaults to the provided application icon.
p_target_url	URL of the page that opens when the user clicks on the push notification. Defaults to the home page of the application. Oracle recommends enabling deep linking or rejoin session on the application for best performance.

Example

The following example sends a push notification to user "SMITH" in application 100.

```
BEGIN
    apex_pwa.send_push_notification (
        p_application_id => 100,
        p_user_name      => 'SMITH',
        p_title          => 'Your order has been shipped',
        p_body            => 'Order #123456 will arrive within 3 days.' );
END;
```

47.5 SUBSCRIBE_PUSH_NOTIFICATIONS Procedure

This procedure subscribes a user to an application to enable receiving push notifications from the application.

Syntax

```
APEX_PWA.SUBSCRIBE_PUSH_NOTIFICATIONS (
    p_application_id      IN NUMBER      DEFAULT [current application id],
    p_user_name           IN VARCHAR2 DEFAULT [current user],
    p_subscription_interface IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	ID of the application that has the push subscription.
p_user_name	Username of the user that has the push subscription.
p_subscription_interface	Subscription object (JSON) generated from a browser.

Example

The following example subscribes a user to push notifications. This is usually used in conjunction with APEX JavaScript API `apex.pwa.subscribePushNotifications` and `apex.pwa.getPushSubscription` that can generate the subscription object.

```
BEGIN
    apex_pwa.subscribe_push_notifications (
        p_subscription_interface => '{ "endpoint": "", "expirationTime": null,
                                         "keys": { "p256dh": "", "auth": "" } }' );
END;
```

47.6 UNSUBSCRIBE_PUSH_NOTIFICATIONS Procedure

This procedure unsubscribes a user from the push notifications of an application.

Syntax

```
APEX_PWA.UNSUBSCRIBE_PUSH_NOTIFICATIONS (
    p_application_id      IN NUMBER  DEFAULT [current application id],
    p_user_name           IN VARCHAR2 DEFAULT [current user],
    p_subscription_interface IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_application_id	ID of the application that has the push subscription.
p_user_name	Username of the user that has the push subscription.
p_subscription_interface	Subscription object (JSON) generated from a browser. If provided, it will only unsubscribe this subscription. If not provided, it will unsubscribe all subscriptions.

Example

The following example unsubscribes a user from push notifications. This is usually used in conjunction with APEX JavaScript API `apex.pwa.unsubscribePushNotifications` and `apex.pwa.getPushSubscription` that can generate the subscription object.

```
BEGIN
    apex_pwa.unsubscribe_push_notifications;
END;
```

48

APEX_REGION

The APEX_REGION package is the public API for handling regions.

- [CLEAR Procedure](#)
- [EXPORT_DATA Function](#)
- [GET_ID Function Signature 1](#)
- [GET_ID Function Signature 2](#)
- [IS_READ_ONLY Function](#)
- [OPEN_QUERY_CONTEXT Function](#)
- [PURGE_CACHE Procedure](#)
- [RESET Procedure](#)

48.1 CLEAR Procedure

This procedure clears region settings (CR and IR pagination, IR report settings).

For interactive report regions, this procedure clears the following settings: control break, aggregate, flashback, chart, number of rows to display, filter, highlight, computation, and group by. However, it does not clear the following: display column list, sorting, report preference (such as view mode, display nulls in detail view, expand/collapse of report settings).

Syntax

```
APEX_REGION.CLEAR (
    p_application_id    IN NUMBER DEFAULT apex_application.g_flow_id,
    p_page_id           IN NUMBER,
    p_region_id          IN NUMBER,
    p_component_id       IN NUMBER DEFAULT NULL );
```

Parameters

Parameter	Description
p_application_id	ID of the application where the region is on.
p_page_id	ID of the page where the region is on.
p_region_id	ID of a specific region.
p_component_id	Region component ID to use. For interactive reports, this is the saved report ID within the current application page.

Example

This example clears the given saved report on application 100, page 1.

```
BEGIN
    APEX_REGION.CLEAR (
```

```
    p_application_id => 100,
    p_page_id         => 1,
    p_region_id       => 2505704029884282,
    p_component_id    => 880629800374638220);
END;
```

48.2 EXPORT_DATA Function

This function exports current region data.

Note:

The APEX_REGION.EXPORT_DATA function only supports native regions at this time.

Syntax

```
APEX_REGION.EXPORT_DATA (
    p_format           IN apex_data_export.t_format,
    --
    p_page_id          IN NUMBER,
    p_region_id        IN NUMBER,
    p_component_id     IN NUMBER             DEFAULT
NULL,
    p_view_mode        IN VARCHAR2          DEFAULT
NULL,
    --
    p_additional_filters IN apex_exec.t_filters      DEFAULT
apex_exec.c_empty_filters,
    --
    p_max_rows         IN NUMBER             DEFAULT
NULL,
    p_parent_column_values IN apex_exec.t_parameters  DEFAULT
apex_exec.c_empty_parameters,
    --
    p_as_clob          IN BOOLEAN            DEFAULT
FALSE,
    --
    p_file_name        IN VARCHAR2          DEFAULT
NULL,
    p_page_size         IN apex_data_export.t_size      DEFAULT
apex_data_export.c_size_letter,
    p_orientation       IN apex_data_export.t_orientation  DEFAULT
apex_data_export.c_orientation_portrait,
    p_data_only         IN BOOLEAN            DEFAULT
FALSE,
    --
    p_pdf_accessible   IN BOOLEAN            DEFAULT
FALSE,
    --
    p_xml_include_declaration IN BOOLEAN          DEFAULT
TRUE )
    RETURN apex_data_export.t_export;
```

Parameters

Parameter	Description
p_format	Export format. Use constants apex_data_export.c_format_*
p_page_id	ID of the page where the region is on.
p_region_id	Open the query context for this specific region ID.
p_component_id	Region component ID to use.
p_view_mode	For Interactive Reports and Interactive Grids, this is the saved report ID within the current application page. For JET charts, use the chart series ID.
p_additional_filters	The view type available for the report. The values can be:
p_max_rows	<ul style="list-style-type: none"> • APEX_IR.C_VIEW_REPORT • APEX_IR.C_VIEW_GROUPBY • APEX_IR.C_VIEW_PIVOT
p_parent_column_values	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 raises.
p_as_clob	Additional filters to apply to the context.
p_file_name	Maximum amount of rows to get. Default unlimited.
p_page_size	For the detail grid in an Interactive Grid Master-Detail relationship. Use this parameter to pass in values for the master-detail parent column(s).
p_orientation	Returns the export contents as a CLOB. Does not work with binary export formats such as PDF and XLSX. Default to false.
p_data_only	Defines the filename of the export.
p_pdf_accessible	Page size of the report. Use constants apex_data_export.c_size_*
p_xml_include_declaration	Orientation of the report page. Use constants apex_data_export.c_orientation_*
	Whether to include column groups, control breaks, aggregates, and highlights.
	Whether to include accessibility tags in the PDF. Defaults to false.
	Whether to include the XML declaration. Defaults to true.

Returns

The export file contents, mime_type, and optionally the report layout.

Examples

Get the export result for a given saved interactive report on page 3 and download as HTML.

```
DECLARE
    l_export      apex_data_export.t_export;
    l_region_id   number;
BEGIN
```

```

SELECT region_id into l_region_id
  FROM apex_application_page_regions
 WHERE application_id = 100
   and page_id = 3
   and static_id = 'classic_report';

l_export := apex_region.export_data (
    p_format      => apex_data_export.c_format_html,
    p_page_id     => 3,
    p_region_id   => l_region_id );

apex_data_export.download( l_export );
END;

```

48.3 GET_ID Function Signature 1

This function gets the region ID based on the dom static ID.

Syntax

```
APEX_REGION.GET_ID (
    p_application_id IN NUMBER  DEFAULT apex.g_flow_id,
    p_page_id        IN NUMBER,
    p_dom_static_id  IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the region.
p_page_id	ID of the page which contains the region.
p_dom_static_id	Static ID of the desired region.

Note:

The p_dom_static_id is not unique and raises a too_many_rows error if multiple regions exist with the same static ID.

Example

The following example gets the ID of the region whose static ID is "my_apex_region" in app 100 on page 1.

```

DECLARE
  l_region_id apex_application_page_regions.region_id%type;
BEGIN
  ...

```

```

l_region_id := apex_region.get_id(
    p_application_id => 100,
    p_page_id        => 1,
    p_dom_static_id  => 'my_apex_region' );
...
END;

```

48.4 GET_ID Function Signature 2

This function gets the region ID based on the region name.

Syntax

```

APEX_REGION.GET_ID (
    p_application_id IN NUMBER  DEFAULT apex.g_flow_id,
    p_page_id        IN NUMBER,
    p_name           IN VARCHAR2 )
RETURN NUMBER;

```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the region.
p_page_id	ID of the page which contains the region.
p_name	Name of the desired region.

Note:

The p_name is not unique and raises a too_many_rows error if multiple regions exist with the same region name.

Example

The following example gets the ID of the region whose name is "Test" in app 100 on page 1.

```

DECLARE
    l_region_id apex_application_page_regions.region_id%type;
BEGIN
    ...
    l_region_id := apex_region.get_id(
        p_application_id => 100,
        p_page_id        => 1,
        p_name           => 'Test' );
    ...
END;

```

48.5 IS_READ_ONLY Function

This function returns TRUE if the current region is rendered read-only and FALSE if not. If the function is called from a context where no region is currently processed, it returns NULL. For example, you can use this function in conditions of a region or its underlying items and buttons.

Syntax

```
FUNCTION IS_READ_ONLY
RETURN BOOLEAN;
```

Parameters

None.

Example

This example returns TRUE if the current region is rendered read-only and FALSE if the region is not rendered read-only.

```
RETURN APEX_REGION.IS_READ_ONLY;
```

48.6 OPEN_QUERY_CONTEXT Function

This function returns an APEX_EXEC query context returning current region data.

This function runs within an autonomous transaction.

Only native regions are supported at this time.

Syntax

```
APEX_REGION.OPEN_QUERY_CONTEXT (
    p_page_id          IN NUMBER,
    p_region_id        IN NUMBER,
    p_component_id     IN NUMBER      DEFAULT NULL,
    p_view_mode        IN VARCHAR2   DEFAULT NULL,
    --
    p_additional_filters IN apex_exec.t_filters DEFAULT
apex_exec.c_empty_filters,
    p_outer_sql         IN VARCHAR2   DEFAULT NULL,
    --
    p_first_row         IN NUMBER      DEFAULT NULL,
    p_max_rows          IN NUMBER      DEFAULT NULL,
    p_total_row_count   IN BOOLEAN     DEFAULT FALSE,
    p_total_row_count_limit IN NUMBER   DEFAULT NULL,
    --
    p_parent_column_values IN apex_exec.t_parameters DEFAULT
apex_exec.c_empty_parameters )
    RETURN apex_exec.t_context;
```

Parameters

Parameter	Description
p_page_id	ID of the page where the region is on.
p_region_id	ID of a specific region to open the query context for.
p_component_id	Region component ID to use. For interactive reports and interactive grids this is the saved report ID within the current application page. For JET charts, use the chart series ID.
p_view_mode	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 the p_view passed does not exist for the current report, an error is raised.
p_additional_filters	Additional filters to apply to the context.
p_outer_sql	Outer SQL query to wrap around the region SQL query. Use #APEX\$SOURCE_DATA# to reference the region source (apex_exec.c_data_source_table_name constant). If this parameter is specified, then the p_columns parameter has no effect. This parameter overrides CHART, GROUP BY or PIVOT views for interactive reports.
p_first_row	Row index to start fetching at. Defaults to 1.
p_max_rows	Maximum amount of rows to get. Default unlimited.
p_total_row_count	Determines whether to retrieve the total row count. Default FALSE. If used together with the p_outer_sql parameter, you must add the APEX\$TOTAL_ROW_COUNT column to the select list of the p_outer_sql query.
p_total_row_count_limit	Upper limit of rows to process the query on. This applies to interactive report aggregations or ordering. Default is no limit.
p_parent_column_values	For the detail grid in an Interactive Grid Master-Detail relationship. Use this parameter to pass in values for the master-detail parent column(s).

Example

The following example demonstrates how to get the query context for a given saved interactive report on page 1 and print the data out as JSON.

```

DECLARE
    l_context apex_exec.t_context;
BEGIN
    l_context := apex_region.open_query_context (
        p_page_id => 1,
        p_region_id => 2505704029884282,
        p_component_id => 880629800374638220 );

    apex_json.open_object;
    apex_json.write_context( 'data', l_context );
    apex_json.close_object;
END;

```

48.7 PURGE_CACHE Procedure

This procedure purges the region cache of the specified application, page, and region.

Syntax

```
APEX_REGION.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

Parameter	Description
p_application_id	ID of the application where the region caches should be purged. Defaults to the current application.
p_page_id	ID of the page where the region caches should be purged. If no value is specified (default), all regions of the application are purged.
p_region_id	ID of a specific region on a page. If no value is specified, all regions of the specified page are purged.
p_current_session_only	Specify true if you only want to purge entries that were saved for the current session. Defaults to FALSE.

Example

This example purges session specific region cache for the whole application.

```
BEGIN
    APEX_REGION.PURGE_CACHE (
        p_current_session_only => true );
END;
```

48.8 RESET Procedure

This procedure resets region settings (such as classic report and interactive report pagination, classic report sort, interactive report and interactive grid report settings, and Region Display Selector tab selection). Only report and Region Display Selector regions are supported at this time.

Syntax

```
APEX_REGION.RESET (
    p_application_id  IN NUMBER  DEFAULT apex_application.g_flow_id,
    p_page_id         IN NUMBER,
    p_region_id       IN NUMBER,
    p_component_id   IN NUMBER  DEFAULT NULL );
```

Parameters

Parameter	Description
p_application_id	ID of the application where the region is on.
p_page_id	ID of the page where the region is on.
p_region_id	ID of a specific region.
p_component_id	Region component ID to use. For interactive reports and interactive grids, this is the saved report ID within the current application page.

Example

This example resets the given saved report on application 100, page 1.

```
BEGIN
    APEX_REGION.RESET (
        p_application_id => 100,
        p_page_id       => 1,
        p_region_id     => 2505704029884282,
        p_component_id  => 880629800374638220);
END;
```

APEX_REST_SOURCE_SYNC

The APEX_REST_SOURCE_SYNC package enables you to synchronize data between tables by merging rows instantly or at scheduled intervals.

- [DISABLE Procedure](#)
- [DYNAMIC_SYNCHRONIZE_DATA Procedure](#)
- [ENABLE Procedure](#)
- [GET_LAST_SYNC_TIMESTAMP Function](#)
- [GET_SYNC_TABLE_DEFINITION_SQL Function](#)
- [IS_RUNNING Function](#)
- [RESCHEDULE Procedure](#)
- [SYNCHRONIZE_DATA Procedure](#)
- [SYNCHRONIZE_TABLE_DEFINITION Procedure](#)

49.1 DISABLE Procedure

This procedure disables automatic synchronization.

Syntax

```
APEX_REST_SOURCE_SYNC.DISABLE (
    p_application_id      IN NUMBER      DEFAULT {current application id},
    p_module_static_id   IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	(Optional) The application ID.
p_module_static_id	Static ID to identify the REST Data Source.

Example

The following example disables synchronization for the `rest_movie` REST Data Source in application 152.

```
BEGIN
    apex_rest_source_sync.disable(
        p_application_id => 152,
        p_module_static_id => 'rest_movie' );
END;
```

49.2 DYNAMIC_SYNCHRONIZE_DATA Procedure

This procedure executes a dynamic data synchronization to the local table based on the provided parameters. The predefined synchronization steps are not executed.

Syntax

```
APEX_REST_SOURCE_SYNC.DYNAMIC_SYNCHRONIZE_DATA (
    p_module_static_id           IN VARCHAR2,
    --
    p_sync_static_id             IN VARCHAR2,
    p_sync_external_filter_expr IN VARCHAR2          DEFAULT NULL,
    p_sync_parameters            IN apex_exec.t_parameters DEFAULT
apex_exec.c_empty_parameters,
    --
    p_application_id             IN NUMBER          DEFAULT
apex_application.g_flow_id );
```

Parameters

Parameter	Description
p_module_static_id	Static ID to identify the REST Data Source.
p_sync_static_id	Static ID for this dynamic synchronization.
p_sync_external_filter_expr	External filter expression to use for this synchronization.
p_sync_parameters	REST Data Source parameters to use for this synchronization.
p_application_id	ID of the application containing the REST Data Source.

Example

The following example performs a dynamic data synchronization with Oracle APEX as the REST Data Source's query parameter.

```
DECLARE
    l_parameters apex_exec.t_parameters;
BEGIN
    apex_exec.add_parameter(
        p_parameters      => l_parameters,
        p_name            => 'query',
        p_value           => 'Oracle APEX' );

    apex_session.create_session(
        p_app_id         => 100,
        p_page_id        => 1,
        p_username        => '...' );

    apex_rest_source_sync.dynamic_synchronize_data(
        p_module_static_id    => 'rest_movie',
        p_sync_static_id      => 'Sync_Oracle_APEX',
        p_sync_parameters     => l_parameters );
END;
```

49.3 ENABLE Procedure

This procedure enables synchronization for the REST Data Source.

Syntax

```
APEX_REST_SOURCE_SYNC.ENABLE (
    p_application_id      IN NUMBER      DEFAULT {current application id},
    p_module_static_id    IN VARCHAR2 )
```

Parameters

Parameter	Description
p_application_id	(Optional) The application ID.
p_module_static_id	Static ID to identify the REST Data Source.

Example

The following example enables synchronization for the `rest_movie` REST Data Source in application 152.

```
BEGIN
    apex_rest_source_sync.enable(
        p_application_id => 152,
        p_module_static_id => 'rest_movie' );
END;
```

49.4 GET_LAST_SYNC_TIMESTAMP Function

This function returns the timestamp of the last successful sync operation.

Syntax

```
APEX_REST_SOURCE_SYNC.GET_LAST_SYNC_TIMESTAMP (
    p_module_static_id  IN VARCHAR2,
    p_application_id    IN NUMBER      DEFAULT {current application id} )
RETURN timestamp with time zone;
```

Parameters

Parameter	Description
p_module_static_id	Static ID to identify the REST Data Source.
p_application_id	ID of the application containing the REST Data Source.

Returns

This function returns the timestamp of the last successful sync operation.

Example

The following example returns the last synchronization timestamp of the "rest_movie" REST Data Source.

```
DECLARE
    l_last_sync_time timestamp with time zone;
BEGIN
    apex_session.create_session(
        p_app_id          => 100,
        p_page_id         => 1,
        p_username        => '...');
    l_last_sync_time := apex_rest_source_sync.get_last_sync_timestamp(
        p_module_static_id => 'rest_movie' );
END;
```

49.5 GET_SYNC_TABLE_DEFINITION_SQL Function

This function generates SQL to synchronize the local table definition with the data profile.

Syntax

```
APEX_REST_SOURCE_SYNC.GET_SYNC_TABLE_DEFINITION_SQL (
    p_module_static_id      IN VARCHAR2,
    p_application_id        IN NUMBER    DEFAULT {current application id},
    p_include_drop_columns  IN BOOLEAN   DEFAULT FALSE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_module_static_id	Static ID to identify the REST Data Source.
p_application_id	(Optional) The application ID.
p_include_drop_columns	If TRUE, generate ALTER TABLE DROP COLUMN statements for columns which do not exist in the data profile any more.

Example

The following example generates the SQL statements (ALTER TABLE) to bring the table in sync with the data profile after the REST Data Source named "rest_movie" has changed.

```
DECLARE
    l_sql varchar2(32767);
BEGIN
    apex_session.create_session(
        p_app_id          => 100,
        p_page_id         => 1,
        p_username        => '...');
    l_sql := apex_rest_source_sync.get_sync_table_definition_sql(
        p_module_static_id => 'rest_movie',
```

```

    p_include_drop_columns  => true );
END;

```

49.6 IS_RUNNING Function

This function determines whether synchronization for the given REST data source is currently running.

Syntax

```
APEX_REST_SOURCE_SYNC.IS_RUNNING (
    p_application_id      IN NUMBER DEFAULT apex_application.g_flow_id,
    p_module_static_id   IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_application_id	ID of the application which contains the automation.
p_module_static_id	Static ID of the automation to disable.

Returns

TRUE if synchronization is currently running. FALSE otherwise.

Example

The following example prints out whether synchronization is currently running.

```
BEGIN
    IF apex_rest_source_sync.is_running(
        p_application_id => 152,
        p_module_static_id => 'rest_movie' )
    THEN
        dbms_output.put_line( 'Synchronization is currently running.' );
    ELSE
        dbms_output.put_line( 'Synchronization is currently not
running.' );
    END IF;
END;
```

49.7 RESCHEDULE Procedure

This procedure sets the next scheduled execution timestamp of the synchronization.

Syntax

```
APEX_REST_SOURCE_SYNC.RESCHEDULE (
    p_application_id      IN NUMBER  DEFAULT apex_application.g_flow_id,
    p_module_static_id   IN VARCHAR2,
    p_next_run_at        IN timestamp with time zone DEFAULT systimestamp );
```

Parameters

Parameter	Description
p_application_id	(Optional): The application ID.
p_module_static_id	Static ID to identify the REST Data Source.
p_next_run_at	Timestamp to execute the next synchronization.

Example

The following example synchronizes the REST Data Source named "rest_movie" immediately.

```
BEGIN
    apex_session.create_session(
        p_app_id      => 100,
        p_page_id     => 1,
        p_username    => '...');
    apex_rest_source_sync.reschedule(
        p_static_id   => 'rest_movie' );
END;
```

49.8 SYNCHRONIZE_DATA Procedure

This procedure executes the configured data synchronization to the local table. The SYNCHRONIZE_DATA procedure requires an APEX session context.

Syntax

```
APEX_REST_SOURCE_SYNC.SYNCHRONIZE_DATA (
    p_module_static_id      IN VARCHAR2,
    p_run_in_background     IN BOOLEAN DEFAULT FALSE,
    p_application_id        IN NUMBER  DEFAULT {current application id} );
```

Parameters

Parameter	Description
p_module_static_id	Static ID to identify the REST Data Source.
p_run_in_background	If TRUE, synchronization runs in the background as a one-time DBMS_SCHEDULER job.
p_application_id	ID of the application containing the REST Data Source.

Example

The following example performs data synchronization immediately, independent of the next scheduled time.

```
BEGIN
    apex_session.create_session(
        p_app_id      => 100,
        p_page_id     => 1,
        p_username    => '...');
```

```
    apex_rest_source_sync.synchronize_data(
        p_module_static_id      => 'rest_movie',
        p_run_in_background     => true );
END;
```

49.9 SYNCHRONIZE_TABLE_DEFINITION Procedure

This procedure synchronizes the local table definition with the data profile.

If the table does not exist, a `CREATE TABLE` statement executes. Table columns are created for visible data profile columns.

If the table already exists, a series of `ALTER TABLE` statements execute in order to align the table with the data profile.

Syntax

```
APEX_REST_SOURCE_SYNC.SYNCHRONIZE_TABLE_DEFINITION (
    p_module_static_id      IN VARCHAR2,
    p_application_id        IN NUMBER    DEFAULT {current application id},
    p_drop_unused_columns   IN BOOLEAN   DEFAULT FALSE );
```

Parameters

Parameter	Description
p_module_static_id	Static ID to identify the REST Data Source.
p_application_id	(Optional) The application ID.
p_drop_unused_columns	If TRUE, the procedure also drops columns which do not exist in the data profile any more.

Example

The following example demonstrates bringing the local synchronization table in sync with the data profile after the REST Data Source named "rest_movie" has changed.

```
BEGIN
    apex_session.create_session(
        p_app_id          => 100,
        p_page_id         => 1,
        p_username        => '...');
    apex_rest_source_sync.synchronize_table_definition(
        p_module_static_id => 'rest_movie',
        p_drop_unused_columns => true );
END;
```

APEX_SEARCH

The APEX_SEARCH package provides search functionality for your applications.

- [QUERY_EXPERT_SEARCH Function](#)
- [QUERY_SEARCH_ENGINE Function](#)
- [SEARCH Function](#)

50.1 QUERY_EXPERT_SEARCH Function

This function converts an end-user search query into the corresponding Oracle Text syntax, enabling advanced and precise searching capabilities.

It processes the search expression and generates a custom search query that can be used with Oracle Text for efficient and accurate text-based searches.

About Search Expression Syntax

The search expression provided as the parameter follows a specific syntax and can include the following elements:

- **Operators** - The search expression can contain specific operators that modify the search behavior:
 - AND - The AND operator is used for combining multiple search terms. For example, "red AND blue" retrieves documents containing both red and blue.
 - OR - The OR operator is mapped to the ACCUM operator of Oracle Text and is used for combining search terms and retrieving documents containing any of the terms. For example, "red OR blue" retrieves documents containing either red or blue with a higher score for the document matching both terms.
 - NOT - The NOT operator is used to exclude specific terms from the search results. For example, "red NOT blue" retrieves documents containing red but not blue.
 - AROUND(d) - The AROUND operator is an abstraction of the NEAR operator in Oracle Text. It is used to find terms within a certain distance (d) of each other. The distance parameter (d) represents the maximum number of words permitted between the two query terms. For example, "red AROUND(3) blue" retrieves documents where red and blue appear within three words or less of each other with a higher score if both terms are closer together.
- **Parentheses** - Parentheses can be used to group search terms and specify the order of evaluation. For example, "(red OR blue) AND green" retrieves documents containing either red or blue and green.
- **Quoted Phrases** - Quoting a phrase (such as "red apple") ensures that the exact phrase is searched for as a whole. For example, "red apple" retrieves documents containing the exact phrase red apple.
- **Fuzzy Prefix** - The syntax enables fuzzy matching using the FUZZY keyword followed by an optional plus (+) or minus (-) sign to increase or decrease the fuzziness. For example, "fuzzy+: red apple" performs a fuzzy match with increased fuzziness for red and apple.

- **Weighted Query Terms** - The search expression supports weighting search terms using the caret symbol (^) followed by a numeric value to indicate the importance or weight of a term. For example, "red^3 apple" assigns a higher weight to the term red compared to apple.

Syntax

```
APEX_SEARCH.QUERY_EXPERT_SEARCH (
    p_search_expression IN VARCHAR2 )
RETURN CLOB;
```

Parameters

Parameter	Description
p_search_expression	End-user search query to convert to Oracle Text syntax. It can include various search operators and keywords.

Returns

This function returns the generated Oracle Text query based on the provided search expression.

Example 1

```
select query_expert_search('(red or white) "summer shorts"') from dual;

TEXT_QUERY
-----
{{red} ACCUM {white}}) AND {summer shorts}
```

Example 2

```
select query_expert_search('fuzzy:- catz dogz') from dual;

TEXT_QUERY
-----
FUZZY({catz},80,100,W) AND FUZZY({dogz},80,100,W)
```

Example 3

```
select query_expert_search('oracle^3 apex') from dual;

TEXT_QUERY
-----
{oracle}*3 AND {apex}
```

50.2 QUERY_SEARCH_ENGINE Function

This function converts a simple end-user search query into the corresponding Oracle Text syntax for a smart search that incorporates query relaxation.

It executes the most restrictive version of a query first (such as exact match search), then progressively relaxes the query using less restrictive queries (such as stem search and fuzzy matching).

While maximizing the number of results, this function ensures that the most exact and relevant matches have a higher score.

Syntax

```
APEX_SEARCH.QUERY_SEARCH_ENGINE (
    p_search_expression IN VARCHAR2 )
    RETURN CLOB;
```

Parameters

Parameter	Description
p_search_expression	End-user search query to convert to Oracle Text syntax.

Returns

This function returns the generated Oracle Text query based on the provided search expression.

Example

```
select query_search_engine('red shorts') from dual;

TEXT_QUERY
-----
<query>
    <textquery>
        <progression>
            <seq>{red} {shorts}</seq>
            <seq>${red} ${shorts}</seq>
            <seq>FUZZY({red},40,1000,W) FUZZY({shorts},40,1000,W)</seq>
            <seq>{red} AND {shorts}</seq>
            <seq>${red} AND ${shorts}</seq>
            <seq>FUZZY({red},40,1000,W) AND FUZZY({shorts},40,1000,W)</seq>
            <seq>{red} ACCUM {shorts}</seq>
            <seq>${red} ACCUM ${shorts}</seq>
            <seq>FUZZY({red},40,1000,W) ACCUM FUZZY({shorts},40,1000,W)</seq>
        </progression>
    </textquery>
</query>
```

50.3 SEARCH Function

This function performs application search.

Syntax

```
APEX_SEARCH.SEARCH (
    p_search_static_ids      IN apex_t_varchar2,
```

```

    p_search_expression      IN VARCHAR2,
    p_apply_order_bys       IN VARCHAR2          DEFAULT 'Y',
    --
    p_return_total_row_count IN VARCHAR2          DEFAULT 'N' )
    RETURN apex_search_result_table pipelined;

```

Parameters

Parameter	Description
p_search_static_ids	List of Search Configuration Static IDs to search within.
p_search_expression	Terms to use in the search.
p_apply_order_bys	Whether to apply the sort settings defined in the search configuration. Pass N in when the query applies its own ORDER BY clause.
p_return_total_row_count	Whether to return the total row count.

Returns

This function returns a table of search results as defined by the `apex_search_result_table` object type. The following columns are available:

CONFIG_LABEL:	Label of the search configuration this result comes from.
RESULT_SEQ:	Sequence of this result within the search configuration.

The following column contents are based on the mapping within the Search Configuration:

PRIMARY_KEY_1:	Primary Key Column 1
PRIMARY_KEY_2:	Primary Key Column 2
TITLE:	Title
SUBTITLE:	Subtitle
DESCRIPTION:	Description
BADGE:	Value to be shown as result "badge"
LAST_MODIFIED:	Timestamp when the result was last modified.
CUSTOM_01:	Custom attribute 1
CUSTOM_02:	Custom attribute 2
CUSTOM_03:	Custom attribute 3
SCORE:	Score or Rank value. If Oracle TEXT is used, the TEXT Score is returned.
LINK:	Link
RESULT_CSS_CLASSES:	Result CSS Classes
FORMATTED_ROW:	Row HTML, if a row template is specified in the search configuration
ICON_TYPE:	Type of the Icon: CLASS, URL, BLOB or INITIALS
ICON_VALUE:	Icon Value, depending on the ICON TYPE
ICON_BLOB:	BLOB containing the icon
ICON_MIMETYPE:	Mimetype of the icon BLOB, if configured
TOTAL_ROW_COUNT:	Total result count, if configured.
CONFIG_ID:	Internal ID of the search configuration this result comes from.

Example

The following example searches for "oracle APEX" within the CUSTOMERS and PRODUCTS search configuration.

```
select config_label, title, subtitle, badge
  from table( apex_search.search(
    p_search_static_ids => apex_t_varchar2( 'PRODUCTS',
'CUSTOMERS' ),
    p_search_expression => 'oracle APEX',
    p_apply_order_bys => 'N' ) );
```

CONFIG_LABEL	TITLE	SUBTITLE	BADGE
Products	APEX vacation app	Subscription Based App	
Products	APEX time entry	On-Premises License	
:			
Customers	John Doe Corp	Software Development	5000
Customers	Development Corp	Software Development	1000
:			

51

APEX_SESSION

The APEX_SESSION package enables you to configure Oracle APEX sessions.

- [ATTACH Procedure](#)
- [CREATE_SESSION Procedure](#)
- [DETACH Procedure](#)
- [DELETE_SESSION Procedure](#)
- [SET_DEBUG Procedure](#)
- [SET_TENANT_ID Procedure](#)
- [SET_TRACE Procedure](#)

51.1 ATTACH Procedure

This procedure sets the environment and runs the Initialization PL/SQL Code based on the given application and current session.

Syntax

```
APEX_SESSION.ATTACH (
    p_app_id      IN NUMBER,
    p_page_id     IN NUMBER,
    p_session_id  IN NUMBER );
```

Parameters

Parameters	Description
p_app_id	The application ID.
p_page_id	The application page.
p_session_id	The session ID.

Raises

- `WWV_FLOW.APP_NOT_FOUND_ERR`: Application does not exist or caller has no access to the workspace.
- `APEX.SESSION.EXPIRED`: Your session has ended.
- `SECURITY_GROUP_ID_INVALID`: Security Group ID (your workspace identity) is invalid.

Example

Attach to session 12345678 for application 100 page 1, then print the app ID and session ID.

```
begin
    apex_session.attach (
```

```
p_app_id      => 100,  
p_page_id     => 1,  
p_session_id  => 12345678 );  
sys.dbms_output.put_line (  
    'App is'||v('APP_ID')||', session is'||v('APP_SESSION'));  
end;
```

See Also:

- [CREATE_SESSION Procedure](#)
- [DELETE_SESSION Procedure](#)
- [DETACH Procedure](#)

51.2 CREATE_SESSION Procedure

This procedure creates a new session for the given application, sets the environment, and runs the application's Initialization PL/SQL Code.

Syntax

```
APEX_SESSION.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

Parameters	Description
p_app_id	The application id.
p_page_id	The application page.
p_username	The session user.
p_call_post_authentication	If true, call post-authentication procedure. The default is false.

Raises

WWV_FLOW.APP_NOT_FOUND_ERR: The application does not exist or the caller has no access to the workspace.

Example

 **Note:**

The `CREATE_SESSION` procedure is not supported in the SQL Commands and SQL Scripts tools within SQL Workshop.

This example creates a session for `EXAMPLE_USER` in application 100 page 1, then prints 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 (
        'App is'||v('APP_ID')||', session is'||v('APP_SESSION'));
end;
```

 **See Also:**

- [DELETE_SESSION Procedure](#)
- [ATTACH Procedure](#)
- [DETACH Procedure](#)

51.3 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

```
APEX_SESSION.DETACH;
```

Example

Detach from the current session.

```
BEGIN
    apex_session.detach;
END;
```

 **See Also:**

- [CREATE_SESSION Procedure](#)
- [DELETE_SESSION Procedure](#)
- [ATTACH Procedure](#)

51.4 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.

Syntax

```
APEX_SESSION.DELETE_SESSION (
    p_session_id    IN NUMBER  DEFAULT apex_application.g_instance );
```

Parameters

Parameters	Description
p_session_id	The session ID.

Raises

- APEX.SESSION.EXPIRED: Your session has ended.
- SECURITY_GROUP_ID_INVALID: Security Group ID (your workspace identity) is invalid.

Example

The following example deletes session 12345678.

```
BEGIN
    apex_session.delete_session (
        p_session_id => 12345678 );
END;
```

 **See Also:**

- [CREATE_SESSION Procedure](#)
- [ATTACH Procedure](#)
- [DETACH Procedure](#)

51.5 SET_DEBUG Procedure

This procedure sets debug level for all future requests in a session.

Syntax

```
APEX_SESSION.SET_DEBUG (
    p_session_id    IN NUMBER DEFAULT apex_application.g_instance,
    p_level         IN apex_debug.t_log_level );
```

Parameters

Parameters	Description
p_session_id	The session ID. Note: The session must belong to the current workspace or the caller must be able to set the session's workspace.
p_level	The debug level. NULL disables debug, 1-9 sets a debug level.

Example 1

This example shows how to set debug for session 1234 to INFO level.

```
apex_session.set_debug (
    p_session_id => 1234,
    p_level      => apex_debug.c_log_level_info );
commit;
```

Example 2

This example shows how to disable debug in session 1234.

```
apex_session.set_debug (
    p_session_id => 1234,
    p_level      => null );
commit;
```

See Also:

- [ENABLE Procedure](#)
- [DISABLE Procedure](#)

51.6 SET_TENANT_ID Procedure

This procedure is used to associate a session with a tenant ID which can be used for building multitenant Oracle APEX applications. Once set, the value of the current tenant can be retrieved using the built-in APP_TENANT_ID.

Syntax

```
APEX_SESSION.SET_TENANT_ID (
    p_tenant_id IN VARCHAR2 );
```

Parameters

Parameter	Description
p_tenant_id	The tenant ID to associate with a session

Raises

PE.DISPLAY_GROUP.SESSION_NOT_VALID: The session doesn't exist.

WWV_FLOW_SESSION_API.TENANT_ID_EXISTS: The tenant ID has already been set.

Example

```
begin
    apex_session.set_tenant_id (
        p_tenant_id => 'ABC');

end;
```

51.7 SET_TRACE Procedure

This procedure sets trace mode in all future requests of a session.

Syntax

```
APEX_SESSION.SET_TRACE (
    p_session_id    IN NUMBER  DEFAULT apex_application.g_instance,
    p_mode          IN VARCHAR2 );
```

Parameters

Parameters	Description
p_session_id	The session ID. The session must belong to the current workspace or the caller must be able to set the session's workspace.
p_level	The trace mode. NULL disables trace, SQL enables SQL trace.

Example 1

This example shows how to enable trace in requests for session 1234.

```
apex_session.set_trace (
    p_session_id => 1234,
        p_mode => 'SQL' );
commit;
```

Example 2

This example shows how to disable trace in requests for session 1234.

```
apex_session.set_trace (
    p_session_id => 1234,
        p_mode => null );
commit;
```

52

APEX_SESSION_STATE

The APEX_SESSION_STATE package encapsulates utilities needed to read and assign session state values.

- [Global Constants](#)
- [Data Types](#)
- [GET_CLOB Function](#)
- [GET_NUMBER Function](#)
- [GET_TIMESTAMP Function](#)
- [GET_VALUE Function](#)
- [GET_VARCHAR2 Function](#)
- [SET_VALUE Procedure Signature 1](#)
- [SET_VALUE Procedure Signature 2](#)
- [SET_VALUE Procedure Signature 3](#)

52.1 Global Constants

The the `t_value` record in the APEX_SESSION_STATE package uses the following data type constants.

```
subtype t_data_type is pls_integer range 1..11;

c_data_type_varchar2      constant t_data_type := apex_exec.c_data_type_varchar2;
c_data_type_clob          constant t_data_type := apex_exec.c_data_type_clob;
```

52.2 Data Types

The APEX_SESSION_STATE package uses the following data types.

The `t_value` record type encapsulates a session state value. Only either `varchar2_value` or `clob_value` is populated at a time.

```
type t_value is record (
    data_type      t_data_type,
    varchar2_value VARCHAR2(32767),
    clob_value     CLOB );
```

52.3 GET_CLOB Function

This function returns the value of a CLOB item identified by `p_item`.

Syntax

```
APEX_SESSION_STATE.GET_CLOB (
    p_item IN VARCHAR2 )
RETURN CLOB;
```

Returns

This function returns the value of the specified item as CLOB.

52.4 GET_NUMBER Function

This function returns the value of a page item identified by `p_item` as NUMBER. This function uses the item's format mask to perform the conversion.

Syntax

```
APEX_SESSION_STATE.GET_NUMBER (
    p_item IN VARCHAR2 )
RETURN NUMBER;
```

Returns

This function returns the value of the specified item as NUMBER.

52.5 GET_TIMESTAMP Function

This function returns the value of a page item identified by `p_item` as TIMESTAMP. This function uses the item's format mask to perform the conversion.

Syntax

```
APEX_SESSION_STATE.GET_TIMESTAMP (
    p_item IN VARCHAR2 )
RETURN TIMESTAMP;
```

Returns

This function returns the value of the specified item as TIMESTAMP.

52.6 GET_VALUE Function

This function returns the value of a page item identified by `p_item` as a generic T_VALUE.

Syntax

```
APEX_SESSION_STATE.GET_VALUE (
    p_item IN VARCHAR2 )
RETURN T_VALUE;
```

Returns

This function returns the value of the specified item as T_VALUE.

52.7 GET_VARCHAR2 Function

This function returns the value of a VARCHAR2 item identified by p_item. This function is the equivalent of the V function. This function raises an exception for values of data type CLOB.

Syntax

```
APEX_SESSION_STATE.GET_VARCHAR2 (
    p_item IN VARCHAR2 )
RETURN VARCHAR2;
```

Returns

This function returns the value of the specified item as VARCHAR2.

52.8 SET_VALUE Procedure Signature 1

This procedure sets an item's session state value based on VARCHAR2.

Syntax

```
APEX_SESSION_STATE.SET_VALUE (
    p_item    IN VARCHAR2,
    p_value   IN VARCHAR2,
    p_commit  IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
p_item	The item name.
p_value	The item value.
p_commit	If TRUE (default), commit after modifying the session state. If FALSE, or if the existing value in session state equals p_value, no commit is issued. This parameter is ignored when the application's Session State Changes attribute is set to "End Of Request."

52.9 SET_VALUE Procedure Signature 2

This procedure sets an item's session state value based on CLOB.

Syntax

```
APEX_SESSION_STATE.SET_VALUE (
    p_item    IN VARCHAR2,
    p_value   IN CLOB,
    p_commit  IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
p_item	The item name.
p_value	The item value.
p_commit	If TRUE (default), commit after modifying the session state. If FALSE, or if the existing value in session state equals p_value, no commit is issued. This parameter is ignored when the application's Session State Changes attribute is set to "End Of Request."

52.10 SET_VALUE Procedure Signature 3

This procedure sets an item's session state value based on a generic t_value.

Syntax

```
APEX_SESSION_STATE.SET_VALUE (
    p_item    IN VARCHAR2,
    p_value   IN t_value,
    p_commit  IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
p_item	The item name.
p_value	The item value.
p_commit	If TRUE (default), commit after modifying the session state. If FALSE, or if the existing value in session state equals p_value, no commit is issued. This parameter is ignored when the application's Session State Changes attribute is set to "End Of Request."

APEX_SHARED_COMPONENT

This package contains APIs to work with shared components. This package is only available in full Oracle APEX installations.

- [Global Constants](#)
- [PUBLISH Procedure](#)
- [REFRESH Procedure](#)

53.1 Global Constants

The APEX_SHARED_COMPONENT package uses the following constants.

```
subtype t_component_type      IS VARCHAR2(15);
subtype t_subscription_status IS VARCHAR2(13);
```

Component Types

c_app_computation	constant t_component_type := 'APP_COMPUTATION';
c_app_item	constant t_component_type := 'APP_ITEM';
c_app_process	constant t_component_type := 'APP_PROCESS';
c_app_setting	constant t_component_type := 'APP_SETTING';
c_authentication	constant t_component_type := 'AUTHENTICATION';
c_authorization	constant t_component_type := 'AUTHORIZATION';
c_build_option	constant t_component_type := 'BUILD_OPTION';
c_component_group	constant t_component_type := 'COMPONENT_GROUP';
c_component_setting	constant t_component_type := 'PLUGIN_SETTINGS';
c_custom_map_bkg	constant t_component_type := 'MAP_BACKGROUND';
c_data_load	constant t_component_type := 'DATA_LOAD';
c_email_template	constant t_component_type := 'EMAIL TEMPLATE';
c_list	constant t_component_type := 'LIST';
c_lov	constant t_component_type := 'LOV';
c_message	constant t_component_type := 'MESSAGE';
c_plugin	constant t_component_type := 'PLUGIN';
c_report_layout	constant t_component_type := 'REPORT_LAYOUT';
c_search_config	constant t_component_type := 'SEARCH_CONFIG';
c_shortcut	constant t_component_type := 'SHORTCUT';
c_user_group	constant t_component_type := 'APP_ACL';
c_rest_data_source	constant t_component_type := 'WEB_SOURCE';
c_duality_view	constant t_component_type := 'DUALITY_VIEW'; -- Duality View
c_json_source	constant t_component_type := 'JSON_SOURCE'; -- JSON Source
c_ai_config	constant t_component_type := 'AI_CONFIG'; -- AI Config

Subscription Status

```
c_status_up_to_date      constant t_subscription_status := 'UP_TO_DATE';
c_status_needs_refresh  constant t_subscription_status := 'NEEDS_REFRESH';
c_status_unknown         constant t_subscription_status := 'UNKNOWN';
```

53.2 PUBLISH Procedure

This procedure publishes a component.

Syntax

```
APEX_SHARED_COMPONENT.PUBLISH (
    p_component_type IN t_component_type,
    p_component_id   IN NUMBER );
```

Parameters

Parameter	Description
p_component_type	Component type to publish.
p_component_id	Component ID to publish.

Example

The following example code publishes a List of Values with component ID 1234.

```
BEGIN
    -- set the current workspace, only required when executing this API
    -- outside
    -- Oracle APEX environment

    apex_util.set_workspace ( p_workspace => 'WORKSPACE_NAME' );

    -- publish a component

    apex_shared_component.publish (
        p_component_type => apex_shared_component.c_lov,
        p_component_id   => 1234 );

    COMMIT;
END;
```

53.3 REFRESH Procedure

This procedure refreshes a component.

Syntax

```
APEX_SHARED_COMPONENT.REFRESH (
    p_component_type IN t_component_type,
    p_component_id    IN NUMBER )
```

Parameters

Parameter	Description
p_component_type	Component type to refresh.
p_component_id	Component ID to refresh.

Example

The following example refreshes all the subscribed components from application 100, which are out of date.

```
BEGIN
    -- set the current workspace, only required when executing this API
    -- outside
    -- APEX environment

    apex_util.set_workspace ( p_workspace => 'WORKSPACE_NAME' );

    -- refresh all the components in an app
    FOR l_component IN (
        SELECT component_type,
               component_id
        FROM apex_subscribed_components
        WHERE application_id      = 100
          AND Subscription_status <>
apex_shared_component.c_status_up_to_date )
    LOOP
        apex_shared_component.refresh (
            p_component_type => l_component.component_type,
            p_component_id   => l_component.component_id );
    END LOOP;

    COMMIT;
END;
```

54

APEX_SPATIAL

This package enables you to use Oracle Locator and the Spatial Option within Oracle APEX.

In an APEX context, the logon user of the database session is typically `APEX_PUBLIC_USER` or `ANONYMOUS`. Spatial developers can not directly use DML on `USER_SDO_GEOM_METADATA` within such a session in SQL Commands within SQL Workshop, for example. The Spatial view's trigger performs DML as the logon user, but it must 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 APEX user. The package also provides a few utilities that simplify the use of Spatial in APEX.

If the `SDO_Geometry` data type is unavailable in the database, then `SPATIAL_IS_AVAILABLE` is the only function within this package, and it returns `FALSE`. All other functions are only available if `SDO_Geometry` is available in the database, and `SPATIAL_IS_AVAILABLE` returns `TRUE`.

- [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](#)
- [SPATIAL_IS_AVAILABLE Function](#)

54.1 Data Types

The `APEX_SPATIAL` package uses the following data types.

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
```

54.2 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

Parameter	Description
p_table_name	Name of the feature table.
p_column_name	Name of the column of type <code>mdsys.sdo_geometry</code> .
p_new_table_name	New name of a feature table (or null, to keep the current value).
p_new_column_name	New name of the column of type <code>mdsys.sdo_geometry</code> (or null, to keep the current value).
p_diminfo	<code>SDO_DIM_ELEMENT</code> array, ordered by dimension, with one entry for each dimension.
p_srid	SRID value for the coordinate system for all geometries in the column.

Example

The code below modifies the dimensions of column `CITIES.SHAPE`.

```
begin
    for l_meta in ( select *
                      from user_sdo_geom_metadata
                     where table_name  = 'CITIES'
                           and column_name = 'SHAPE' )
    loop
        apex_spatial.change_geom_metadata (
            p_table_name  => l_meta.table_name,
            p_column_name => l_meta.column_name,
            p_diminfo     => SDO_DIM_ARRAY (
                SDO_DIM_ELEMENT('X', -180, 180, 0.1),
                SDO_DIM_ELEMENT('Y', -90, 90, 0.1) ),
            p_srid        => l_meta.srid );
    end loop;
end;
```

54.3 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

Parameter	Description
p_lon	Longitude position of the lower left point.
p_lat	Latitude position of the lower left point.
p_radius	Radius of the circle in meters.
p_arc_tolerance	Arc tolerance (default 20).
p_srid	Reference system (default c_wgs_84).

Returns

Return	Description
mdsys.sdo_geometry	The geometry for the polygon that approximates the circle.

Example

This example is a query that returns a polygon that approximates a circle at (0, 0) with radius 1.

```
select apex_spatial.circle_polygon(0, 0, 1) from dual
```

54.4 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

Parameter	Description
p_table_name	Name of the feature table.
p_column_name	Name of the column of type mdsys.sdo_geometry.
p_drop_index	If TRUE (default is FALSE), drop the spatial index on the column.

Example

This example deletes metadata on column CITIES.SHPE 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;
```

54.5 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

Parameter	Description
p_table_name	The name of the feature table.
p_column_name	The name of the column of type mdsys.sdo_geometry.
p_diminfo	The SDO_DIM_ELEMENT array, ordered by dimension, with one entry for each dimension.
p_srid	The SRID value for the coordinate system for all geometries in the column.
p_create_index_name	If not null, a spatial index on the column is created with this name. Only simple column names are supported, function based indexes or indexes on object attributes cause an error. For more complex requirements, leave this parameter null (the default) and manually create the index.

Example

This example creates table CITIES, spatial metadata and an index on column CITIES.SHPE.

```
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
/

```

54.6 INSERT_GEOM_METADATA_LONLAT Procedure

This procedure inserts a spatial metadata record that is suitable for longitude/latitude and optionally creates a spatial index.

Syntax

```
APEX_SPATIAL.INSERT_GEOM_METADATA_LONLAT (
    p_table_name      IN VARCHAR2,
    p_column_name     IN VARCHAR2,
    p_tolerance       IN NUMBER DEFAULT 1,
    p_create_index_name IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_table_name	Name of the feature table.
p_column_name	Name of the column of type mdsys.sdo_geometry.
p_tolerance	Tolerance value in each dimension, in meters (default 1).
p_create_index_name	If not null, a spatial index on the column is created with this name. Only simple column names are supported, function based indexes or indexes on object attributes cause an error. For more complex requirements, leave this parameter null (the default) and manually create the index.

Example

The code below creates table CITIES and spatial metadata for the column CITIES.SHAPe. By passing CITIES_IDX_SHAPE to p_create_index_name, the API call automatically creates an index on the spatial column.

```

create table cities (
    city_id    number primary key,
    city_name  varchar2(30),
    shape      mdsys.sdo_geometry )
/
begin
    apex_spatial.insert_geom_metadata_lonlat (
        p_table_name      => 'CITIES',
        p_column_name     => 'SHAPE',
        p_create_index_name => 'CITIES_IDX_SHAPE' );

```

```
end;  
/
```

54.7 POINT Function

This function creates a point at (p_lon, p_lat).

Syntax

```
APEX_SPATIAL.POINT (  
    p_lon      IN NUMBER,  
    p_lat      IN NUMBER,  
    p_srid     IN t_srid DEFAULT c_wgs_84 )  
RETURN mdsys.sdo_geometry;
```

Parameters

Parameter	Description
p_lon	Longitude position.
p_lat	Latitude position.
p_srid	Reference system (default c_wgs_84).

Returns

Return	Description
mdsys.sdo_geometry	The geometry for the point.

Example

This example is a query that returns a point at (10, 50).

```
select apex_spatial.point(10, 50) from dual;
```

This example is equivalent to:

```
select mdsys.sdo_geometry(2001, 4326, sdo_point_type(10, 50, null), null, null)  
from dual;
```

54.8 RECTANGLE Function

This function creates a rectangle from point at (p_lon1, p_lat1) to (p_lon2, p_lat2).

Syntax

```
APEX_SPATIAL.RECTANGLE (  
    p_lon1      IN NUMBER,  
    p_lat1      IN NUMBER,  
    p_lon2      IN NUMBER,  
    p_lat2      IN NUMBER,
```

```
p_srid      IN t_srid DEFAULT c_wgs_84 )
RETURN mdsys.sdo_geometry;
```

Parameters

Parameter	Description
p_lon1	Longitude position of the lower left point.
p_lat1	Latitude position of the lower left point.
p_lon2	Longitude position of the upper right point.
p_lat2	Latitude position of the upper right point.
p_srid	Reference system (default c_wgs_84).

Returns

Return	Description
mdsys.sdo_geometry	The geometry for the rectangle (p_lon1, p_lon2, p_lat2, p_lat1).

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;
```

54.9 SPATIAL_IS_AVAILABLE Function

This function returns whether spatial is available in the database.

Syntax

```
APEX_SPATIAL.SPATIAL_IS_AVAILABLE (
    spatial_is_available )
RETURN BOOLEAN;
```

Returns

Parameter	Description
*	True when spatial (SDO_GEOGRAPHY) is available in the database. Otherwise, false.

APEX_STRING

The APEX_STRING package provides utilities for the following data types:

- apex_t_clob
- apex_t_number
- apex_t_varchar2
- clob
- varchar2

Unless otherwise noted, the APIs expect arrays to be continuous (that is, without holes that `coll.delete(n)` operations generate).

- [FORMAT Function](#)
- [GET_INITIALS Function](#)
- [GET_SEARCHABLE_PHRASES Function](#)
- [GREP Function Signature 1](#)
- [GREP Function Signature 2](#)
- [GREP Function Signature 3](#)
- [INDEX_OF Function Signature 1](#)
- [INDEX_OF Function Signature 2](#)
- [JOIN_CLOB Function](#)
- [JOIN_CLOBS Function](#)
- [JOIN Function Signature 1](#)
- [JOIN Function Signature 2](#)
- [NEXT_CHUNK Function](#)
- [PLIST_DELETE Procedure](#)
- [PLIST_EXISTS Function](#)
- [PLIST_GET Function](#)
- [PLIST_GET_KEY Function](#)
- [PLIST_PUSH Procedure](#)
- [PLIST_PUT Function](#)
- [PLIST_TO_JSON_CLOB Function](#)
- [PUSH Procedure Signature 1](#)
- [PUSH Procedure Signature 2](#)
- [PUSH Procedure Signature 3](#)
- [PUSH Procedure Signature 4](#)

- PUSH Procedure Signature 5
- PUSH Procedure Signature 6
- PUSH Procedure Signature 7
- SHUFFLE Function
- SHUFFLE Procedure
- SPLIT Function Signature 1
- SPLIT Function Signature 2
- SPLIT_CLOBS Function
- SPLIT_NUMBERS Function
- STRING_TO_TABLE Function
- TABLE_TO_CLOB Function
- TABLE_TO_STRING Function

55.1 FORMAT Function

This function returns a formatted string with substitutions applied.

Returns `p_message` after replacing each `<n>`th occurrence of `%s` with `p<n>` and each occurrence of `%<n>` with `p<n>`. If `p_max_length` is not null, `substr(p<n>,1,p_arg_max_length)` is used instead of `p<n>`.

Use `%%` in `p_message` to emit a single % character. Use `%n` to emit a newline.

Syntax

```
APEX_STRING.FORMAT (
    p_message      IN VARCHAR2,
    p0             IN VARCHAR2      DEFAULT NULL,
    p1             IN VARCHAR2      DEFAULT NULL,
    p2             IN VARCHAR2      DEFAULT NULL,
    p3             IN VARCHAR2      DEFAULT NULL,
    p4             IN VARCHAR2      DEFAULT NULL,
    p5             IN VARCHAR2      DEFAULT NULL,
    p6             IN VARCHAR2      DEFAULT NULL,
    p7             IN VARCHAR2      DEFAULT NULL,
    p8             IN VARCHAR2      DEFAULT NULL,
    p9             IN VARCHAR2      DEFAULT NULL,
    p10            IN VARCHAR2      DEFAULT NULL,
    p11            IN VARCHAR2      DEFAULT NULL,
    p12            IN VARCHAR2      DEFAULT NULL,
    p13            IN VARCHAR2      DEFAULT NULL,
    p14            IN VARCHAR2      DEFAULT NULL,
    p15            IN VARCHAR2      DEFAULT NULL,
    p16            IN VARCHAR2      DEFAULT NULL,
    p17            IN VARCHAR2      DEFAULT NULL,
    p18            IN VARCHAR2      DEFAULT NULL,
    p19            IN VARCHAR2      DEFAULT NULL,
    p_max_length   IN PLS_INTEGER  DEFAULT 1000,
    p_prefix        IN VARCHAR2      DEFAULT NULL )
    return VARCHAR2
```

Parameters

Parameters	Description
p_message	Message string with substitution placeholders.
p0-p19	Substitution parameters.
p_max_length	If not null (default is 1000), cap each p<n> at p_max_length characters. The tilde (~) character is appended to indicate that the original value exceeded this length.
p_prefix	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.

Example 1

```
APEX_STRING.FORMAT('%s+%s=%s', 1, 2, 'three')
-> 1+2=three
```

```
APEX_STRING.FORMAT('%1+%2=%0', 'three', 1, 2)
-> 1+2=three
```

Example 2

```
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;
```

55.2 GET_INITIALS Function

Returns the initials of the words in a string. Splits the string into words and takes the first letter of each word. If there is only one word, then it takes the first p_cnt letters.

Syntax

```
APEX_STRING.GET_INITIALS (
  p_str IN VARCHAR2,
  p_cnt IN NUMBER DEFAULT 2 )
RETURN VARCHAR2
```

Parameters

Parameters	Description
p_string	The input string.
p_cnt	The maximum length of the initials to return. The default is 2.

Example

Get initials from "John Doe".

```
begin
  sys.dbms_output.put_line(apex_string.get_initials('John Doe'));
end;
-> JD

begin
  sys.dbms_output.put_line(apex_string.get_initials(p_str => 'Andres Homero
Lozano Garza', p_cnt => 3));
end;
-> AHL
```

55.3 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

```
FUNCTION GET_SEARCHABLE_PHRASES (
  p_strings  IN  apex_t_varchar2,
  p_max_words IN  PLS_INTEGER      DEFAULT 3,
  p_language  IN  apex_t_varchar2 DEFAULT 'en' )
  RETURN apex_t_varchar2;
```

Parameters

Parameters	Description
p_string	The input string.
p_max_words	The maximum number of words in a phrase. The default is 3.

Parameters	Description
p_language	The language identifier for stopwords, defaults to "en". Supported values are "cn", "de", "en", "es", "fr", "it", "ja", "ko", "pt-br".

Example

Prints keywords in the given input string.

```
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;
-> oracle:oracle apex:oracle apex 19.1:apex:apex 19.1:19.1:great:low:low
code:code
```

55.4 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

Parameters	Description
p_table	The input table.
p_pattern	The regular expression.
p_modifier	The regular expression modifier.
p_subexpression	The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If > 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.
p_limit	Limitation for the number of elements in the return table. If null (the default), there is no limit.

Example

Collect and print basenames of sql files in input collection.

```
declare
    l_sqlfiles apex_t_varchar2;
begin
    l_sqlfiles := apex_string.grep (
        p_table => apex_t_varchar2('a.html','b.sql', 'C.SQL'),
        p_pattern => '(\w+)\.sql',
        p_modifier => 'i',
        p_subexpression => '1' );
    sys.dbms_output.put_line(apex_string.join(l_sqlfiles, ':'));
end;
-> b:C
```

55.5 GREP Function Signature 2

Returns the values of the input `varchar2` that match a regular expression.

Syntax

```
APEX_STRING.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

Parameters	Description
<code>p_str</code>	The input <code>varchar2</code> .
<code>p_pattern</code>	The regular expression.
<code>p_modifier</code>	The regular expression modifier.
<code>p_subexpression</code>	The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If > 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.
<code>p_limit</code>	Limitation for the number of elements in the return table. If null (the default), there is no limit.

Example

Collect and print key=value definitions.

```
declare
    l plist apex_t_varchar2;
begin
    l plist := apex_string.grep (
        p_str => 'define k1=v1'||chr(10)||
```

```

        'define k2 = v2',
p_pattern => 'define\s+(\w+)\s*=\s*([^\n|\chr(10)|\n]*',
p_modifier => 'i',
p_subexpression => '1,2' );
sys.dbms_output.put_line(apex_string.join(l plist, ':'));
end;
-> k1:v1:k2:v2

```

55.6 GREP Function Signature 3

Returns the values of the input `clob` that match a regular expression.

Syntax

```

APEX_STRING.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

Parameters	Description
p_str	The input <code>clob</code> .
p_pattern	The regular expression.
p_modifier	The regular expression modifier.
p_subexpression	The subexpression which should be returned. If null, return the complete table value. If 0 (the default), return the matched expression. If > 0, return the subexpression value. You can also pass a comma separated list of numbers, to get multiple subexpressions in the result.
p_limit	Limitation for the number of elements in the return table. If null (the default), there is no limit.

Example

Collect and print key=value definitions.

```

declare
    l plist apex_t_varchar2;
begin
    l plist := apex_string.grep (
        p_str => to_clob('define k1=v1||chr(10)||'
                           'define k2 = v2',
        p_pattern => 'define\s+(\w+)\s*=\s*([^\n|\chr(10)|\n]*',
chr(10)||\n)*',
        p_modifier => 'i',
        p_subexpression => '1,2' );
    sys.dbms_output.put_line(apex_string.join(l plist, ':'));

```

```
end;
-> k1:v1:k2:v2
```

55.7 INDEX_OF Function Signature 1

This function returns the first position in the list where `p_value` is stored. If not found, returns NULL.

Syntax

```
APEX_STRING.INDEX_OF (
    p_table IN wwv_flow_t_varchar2,
    p_value IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

Parameter	Description
<code>p_table</code>	The table.
<code>p_value</code>	Value that is being searched for.

Returns

Index of the searched value in the table.

Example

The following example prints the index of the given input string in the table.

```
BEGIN
    sys.dbms_output.put_line (
        apex_string.index_of (
            p_table => apex_t_varchar2 (
                'Dog',
                'Cat',
                'Capybara' ),
            p_value => 'Capybara' ) );
END;
-> 3
```

55.8 INDEX_OF Function Signature 2

This function returns the first position in the list where `p_value` is stored. If not found, returns NULL.

Syntax

```
APEX_STRING.INDEX_OF (
    p_table IN wwv_flow_global_vc_arr2,
    p_value IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_table	The table.
p_value	Value that is being searched for.

Returns

Index of the searched value in the table.

Example

The following example prints the index of the given input string in the table.

```
DECLARE
    l_list      apex_application_global.vc_arr2;
BEGIN
    l_list(1) := 'Dog';
    l_list(2) := 'Capybara';
    l_list(3) := 'Cat';
    sys.dbms_output.put_line (
        apex_string.index_of (
            p_table => l_list,
            p_value => 'Capybara' ) );
END;
-> 3
```

55.9 JOIN_CLOB Function

Returns the values of the `apex_t_varchar2` input table `p_table` as a concatenated clob, separated by `p_sep`.

Syntax

```
APEX_STRING.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

Parameters	Description
p_table	The input table.
p_sep	The separator, default is line feed.
p_dur	The duration of the clob, default <code>sys.dbms_lob.call</code>

Example

Concatenate numbers, separated by ':'.

```
apex_string.join_clob(apex_t_varchar2('1','2','3'),':')
-> 1:2:3
```

55.10 JOIN_CLOBS Function

This function returns the values of the `apex_t_clob` input table `p_table` as a concatenated clob, separated by `p_sep`.

Syntax

```
APEX_STRING.JOIN_CLOBS (
    p_table IN apex_t_clob,
    p_sep    IN VARCHAR2      DEFAULT apex_application.LF,
    p_dur    IN PLS_INTEGER   DEFAULT sys.dbms_lob.call )
RETURN CLOB;
```

Parameters

Parameter	Description
<code>p_table</code>	The input table.
<code>p_sep</code>	The separator, default is line feed.
<code>p_dur</code>	The duration of the clob, default <code>sys.dbms_lob.call</code>

Example

The following example concatenates numbers, separated by ':'.

```
apex_string.join_clobs(apex_t_clob('1','2','3'),':')
-> 1:2:3
```

55.11 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

```
APEX_STRING.JOIN (
    p_table IN apex_t_varchar2,
    p_sep    IN VARCHAR2 DEFAULT apex_application.LF )
RETURN VARCHAR2
```

Parameters

Parameters	Description
p_table	The input table.
p_sep	The separator, default is line feed.

Example

Concatenate numbers, separated by ':'.

```
apex_string.join(apex_t_varchar2('a','b','c'),':')
-> a:b:c
```

55.12 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

```
APEX_STRING.JOIN (
    p_table IN apex_t_number,
    p_sep IN VARCHAR2 DEFAULT apex_application.LF )
RETURN VARCHAR2
```

Parameters

Parameters	Description
p_table	The input table.
p_sep	The separator, default is line feed.

Example

Concatenate numbers, separated by ':'.

```
apex_string.join(apex_t_number(1,2,3),':')
-> 1:2:3
```

55.13 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

```
APEX_STRING.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

Parameters	Description
p_str	The input clob.
p_chunk	The chunk value (in/out).
p_offset	The position in p_str, where the next chunk should be read from (in/out).
p_amount	The amount of characters that should be read (default 8191).

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

```

55.14 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

Parameters	Description
p_table	The input table.
p_key	The input key.

Raised Errors

Parameters	Description
NO_DATA_FOUND	Given key does not exist in table.

Example

Remove value of property "key2".

```
declare
    l plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
begin
    apex_string.plist_delete(l plist,'key2');
    sys.dbms_output.put_line(apex_string.join(l plist,':'));
end;
-> key1:foo
```

55.15 PLIST_EXISTS Function

Returns whether a key exists in the property list.

Syntax

```
APEX_STRING.PLIST_EXISTS (
    p_table IN apex_t_varchar2,
    p_key   IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_table	The input table.
p_key	The input key.

Raises

NO_DATA_FOUND: Given key does not exist in table.

Example

The following example prints whether properties key1, key2 and key3 exist.

```
DECLARE
    l plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
```

```

BEGIN
    FOR i IN 1 .. 3 LOOP
        sys.dbms_output.put_line(
            'key'||i||': '|||
            CASE apex_string.plist_exists(l plist,'key'||i)
            WHEN true then 'TRUE'
            ELSE 'FALSE'
            END);
    END LOOP;
END;
-> key1:TRUE
key2:TRUE
key3:FALSE

```

55.16 PLIST_GET Function

This function gets the property list value for a key.

Syntax

```
APEX_STRING.PLIST_GET (
    p_table IN apex_t_varchar2,
    p_key IN VARCHAR2 )
RETURN VARCHAR2
```

Parameters

Parameters	Description
p_table	The input table.
p_key	The input key.

Raised Errors

Parameters	Description
NO_DATA_FOUND	Given key does not exist in table.

Example

Get value of property "key2".

```

declare
    l plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
begin
    sys.dbms_output.put_line(apex_string.plist_get(l plist,'key2'));
end;
-> bar

```

55.17 PLIST_GET_KEY Function

Gets the first property list key that maps to a given value. Returns NULL if the value can not be found.

Syntax

```
APEX_STRING.PLIST_GET_KEY (
    p_table IN apex_t_varchar2,
    p_value IN VARCHAR2 )
RETURN varchar2;
```

Parameters

Parameter	Description
p_table	The input table.
p_value	The input value.

Example

The following example gets key of property "bar."

```
DECLARE
    l_plist apex_t_varchar2 := apex_t_varchar2('key1','foo','key2','bar');
BEGIN
    sys.dbms_output.put_line(apex_string.plist_get_key(l_plist,'bar'));
END;
-> key2
```

55.18 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 apex_t_varchar2,
    p_key   IN VARCHAR2,
    p_value IN VARCHAR2 );
```

Parameters

Parameters	Description
p_table	The input table.
p_key	The input key.
p_value	The input value.

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;
-> bar

```

55.19 PLIST_PUT Function

This function inserts or updates property list value for a key.

Syntax

```

PLIST_PUT (
    p_table IN OUT NOCOPY apex_t_varchar2,
    p_key   IN          VARCHAR2,
    p_value IN          VARCHAR2 );

```

Parameters

Parameters	Description
p_table	The input table.
p_key	The input key.
p_value	The input value.

Example

Set property value to "key2".

```

declare
    l plist apex_t_varchar2 := apex_t_varchar2('key1','foo');
begin
    apex_string.plist_put(l plist,'key2','bar');
    sys.dbms_output.put_line(apex_string.plist_get(l plist,'key2'));
end;
-> bar

```

55.20 PLIST_TO_JSON_CLOB Function

This function converts a `apex_t_varchar2` record to a `sys.json_object_t` object type and stringifies it.

Elements with odd numbers are the attribute names.

Elements with even numbers are the attribute values.

Syntax

```

APEX_STRING.PLIST_TO_JSON_CLOB (
    p plist IN apex_t_varchar2 )
RETURN CLOB;

```

Parameters

Parameter	Description
p plist	The table.

Returns

CLOB containing a JSON object with keys and values of the given p plist.

Example

The following example creates the JSON object {"key1":"foo","key2":"bar"}

```
DECLARE
    l_attributes apex_application_page_regions.attributes%type;
BEGIN
    l_attributes := apex_string.plist_to_json_clob(apex_t_varchar2(
        'key1', 'foo',
        'key2', 'bar' ));
    dbms_output.put_line(l_attributes);
END;
```

55.21 PUSH Procedure Signature 1

This procedure appends value to apex_t_varchar2 table.

Syntax

```
APEX_STRING.PUSH (
    p_table IN OUT NOCOPY apex_t_varchar2,
    p_value IN VARCHAR2 );
```

Parameters

Parameter	Description
p_table	Defines the table.
p_value	Specifies the value to be added.

Example

The following example demonstrates how to append 2 values, then prints the table.

```
DECLARE
    l_table apex_t_varchar2;
BEGIN
    apex_string.push(l_table, 'a');
    apex_string.push(l_table, 'b');
    sys.dbms_output.put_line(apex_string.join(l_table, ':'));
END;
-> a:b
```

55.22 PUSH Procedure Signature 2

This procedure appends a value to `apex_t_clob` table.

Syntax

```
APEX_STRING.PUSH (
    p_table IN OUT NOCOPY apex_t_clob,
    p_value IN             apex_t_clob );
```

Parameters

Parameter	Description
<code>p_table</code>	Defines the table.
<code>p_value</code>	Specifies the value to be added.

Example

The following example demonstrates how to append two values, then prints the table.

```
DECLARE
    l_table apex_t_clob;
    l_clob clob;
BEGIN
    apex_string.push(l_table, 'a');
    l_clob := 'large quantity of text...';
    apex_string.push(l_table, l_clob);
    sys.dbms_output.put_line(apex_string.join_clobs(l_table, ':'));
END;
-> a:large quantity of text...
```

55.23 PUSH Procedure Signature 3

This procedure appends a value to `apex_t_number` table.

Syntax

```
APEX_STRING.PUSH (
    p_table IN OUT NOCOPY apex_t_number,
    p_value IN             NUMBER );
```

Parameters

Parameter	Description
<code>p_table</code>	Defines the table.
<code>p_value</code>	Specifies the value to be added.

Example

The following example demonstrates how to append two values, then prints the table.

```
DECLARE
    l_table apex_t_number;
BEGIN
    apex_string.push(l_table, 1);
    apex_string.push(l_table, 2);
    sys.dbms_output.put_line(apex_string.join(l_table, ':'));
END;
-> 1:2
```

55.24 PUSH Procedure Signature 4

This procedure appends collection values to `apex_t_varchar2` table.

Syntax

```
APEX_STRING. PUSH (
    p_table  IN OUT NOCOPY apex_t_varchar2,
    p_values IN           apex_t_varchar2 );
```

Parameters

Parameter	Description
<code>p_table</code>	Defines the table.
<code>p_values</code>	Specifies the values that should be added to <code>p_table</code> .

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
```

55.25 PUSH Procedure Signature 5

This procedure appends collection values to the `apex_t_clob` table.

Syntax

```
APEX_STRING.PUSH (
    p_table  IN OUT NOCOPY apex_t_clob,
    p_values IN                 apex_t_clob )
```

Parameters

Parameter	Description
p_table	The table.
p_values	Values to be added to p_table.

Example

The following example appends single value and multiple values, then prints the table.

```
DECLARE
    l_table apex_t_clob;
BEGIN
    apex_string.push(l_table, 'a');
    apex_string.push(l_table, apex_t_clob('1','2','3'));
    sys.dbms_output.put_line(apex_string.join_clobs(l_table, ':'));
END;
-> a:1:2:3
```

55.26 PUSH Procedure Signature 6

This procedure appends values of a PL/SQL table to the apex_t_varchar2 table.

Syntax

```
APEX_STRING.PUSH (
    p_table  IN OUT NOCOPY apex_t_varchar2,
    p_values IN                 apex_application_global_vc_arr2 )
```

Parameters

Parameter	Description
p_table	The table.
p_values	Values to add to p_table.

Example

The following example appends then prints the table.

```
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

```

55.27 PUSH Procedure Signature 7

This procedure appends number collection values to the `apex_t_varchar2` table.

Syntax

```

APEX_STRING.PUSH (
    p_table      IN OUT NOCOPY wwv_flow_t_varchar2,
    p_values     IN          wwv_flow_t_number,
    p_format_mask IN          VARCHAR2 DEFAULT NULL );

```

Parameters

Parameter	Description
<code>p_table</code>	The table.
<code>p_values</code>	Values that should be added to <code>p_table</code> .
<code>p_format_mask</code>	Format mask to use when converting numbers to strings.

Example

The following example appends 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_number(1,2,3), 'FM990D00');
    sys.dbms_output.put_line(apex_string.join(l_table, ':'));
END;
-> a:1.00:2.00:3.00

```

55.28 SHUFFLE Function

Returns the input table values, re-ordered.

Syntax

```

APEX_STRING.SHUFFLE (
    p_table IN apex_t_varchar2 )
RETURN apex_t_varchar2;

```

Parameters

Parameters	Description
p_table	The input table.

Example

Shuffle and print l_table.

```
declare
    l_table apex_t_varchar2 := apex_string.split('1234567890',null);
begin
    sys.dbms_output.put_line(apex_string.join(apex_string.shuffle(l_table),':'));
end;
-> a permutation of 1:2:3:4:5:6:7:8:9:0
```

55.29 SHUFFLE Procedure

This procedure randomly re-orders the values of the input table.

Syntax

```
APEX_STRING.SHUFFLE (
    p_table IN OUT NOCOPY apex_t_varchar2 );
```

Parameters

Parameters	Description
p_table	The input table, which will be modified by the procedure.

Example

Shuffle and print l_table.

```
declare
    l_table apex_t_varchar2 := apex_string.split('1234567890',null);
begin
    apex_string.shuffle(l_table);
    sys.dbms_output.put_line(apex_string.join(l_table,:));
end;
-> a permutation of 1:2:3:4:5:6:7:8:9:0
```

55.30 SPLIT Function Signature 1

Use this function to split input string at separator.

Syntax

```
APEX_STRING.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

Parameters	Description
p_str	The input string.
p_sep	The separator. Splits at line feed by default. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression (max 512 characters).
p_limit	Maximum number of splits, ignored if null. If smaller than the total possible number of splits, the last table element contains the rest.

Examples

```
apex_string.split(1||chr(10)||2||chr(10)||3)
-> apex_t_varchar2('1','2','3')

apex_string.split('1:2:3',':')
-> apex_t_varchar2('1','2','3')

apex_string.split('123',null)
-> apex_t_varchar2('1','2','3')

apex_string.split('1:2:3:4',':',2)
-> apex_t_varchar2('1','2:3:4')

apex_string.split('key1=val1, key2=val2','\s*[=,\]\s*')
-> apex_t_varchar2('key1','val1','key2','val2')
```

55.31 SPLIT Function Signature 2

Use this function to split input clob at separator.

Syntax

```
APEX_STRING.SPLIT (
    p_str IN CLOB,
    p_sep IN VARCHAR2 DEFAULT apex_application.LF )
RETURN apex_t_varchar2;
```

Parameters

Parameters	Description
p_str	The input clob.

Parameters	Description
p_sep	The separator. Splits at line feed by default. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression (max 512 characters).

Example

```
apex_string.split('1:2:3',':')
-> apex_t_varchar2('1','2','3')
```

55.32 SPLIT_CLOBS Function

This function splits input clobs at the separator and returns a table of clobs.

Syntax

```
APEX_STRING.SPLIT_CLOBS (
    p_str    IN CLOB,
    p_sep    IN VARCHAR2      DEFAULT apex_application.LF,
    p_limit  IN PLS_INTEGER  DEFAULT NULL )
RETURN apex_t_clob;
```

Parameters

Parameter	Description
p_str	The input blob.
p_sep	The separator. Splits at line feed by default. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression (max 512 characters).
p_limit	Maximum number of splits. Ignored if null. If smaller than the total possible number of splits, the last table element contains the rest.

Example

```
apex_string.split_clob('1:2:3',':')
-> apex_t_clob('1','2','3')
```

55.33 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

Parameters	Description
p_str	The input varchar2.
p_sep	The separator. Splits at line feed by default. If null, split after each character. If a single character, split at this character. If more than 1 character, split at regular expression (max 512 characters).

Example

```
apex_string.split_numbers('1:2:3',':')
-> apex_t_number(1,2,3)
```

55.34 STRING_TO_TABLE Function

This function returns the split input at separator, returning a `vc_arr2`.

Syntax

```
APEX_STRING.STRING_TO_TABLE (
    p_str    IN VARCHAR2,
    p_sep    IN VARCHAR2 DEFAULT ':' )
RETURN apex_application_global_vc_arr2;
```

Parameters

Parameters	Description
p_str	The input varchar2.
p_sep	The separator, no regexp or split at char. Defaults to ':'.

Example

```
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
```

55.35 TABLE_TO_CLOB Function

This function returns the values of the `apex_application_global_vc_arr2` input table `p_table` as a concatenated clob, separated by `p_sep`.

Syntax

```
APEX_STRING.TABLE_TO_CLOB (
    p_table IN apex_application_global.vc_arr2,
    p_sep    IN VARCHAR2              DEFAULT wwv_flow.LF,
    p_dur    IN PLS_INTEGER          DEFAULT sys.dbms_lob.call )
RETURN CLOB;
```

Parameters

Parameter	Description
p_table	The input table.
p_sep	The separator. Default is line feed.
p_dur	The duration of the clob. Default sys.dbms_lob.call.

Example

The following example concatenates numbers, separated by ':'

```
DECLARE
    l_table apex_application_global.vc_arr2;
BEGIN
    l_table(1) := '1';
    l_table(2) := '2';
    l_table(3) := '3';

    sys.dbms_output.put_line(apex_string.table_to_clob(l_table, ':'));
END;
-> 1:2:3
```

55.36 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

```
FUNCTION TABLE_TO_STRING (
    p_table IN apex_application_global.vc_arr2,
    p_sep   IN VARCHAR2              DEFAULT ':' )
RETURN VARCHAR2;
```

Parameters

Parameters	Description
p_table	The input table, assumes no holes and index starts at 1.
p_sep	The separator, default is ':'.

Example

Concatenate numbers, separated by ':'.

```
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
```

APEX_STRING_UTIL

The APEX_STRING_UTIL package provides additional string related utilities.

- DIFF Function
- FIND_EMAIL_ADDRESSES Function
- FIND_EMAIL_FROM Function
- FIND_EMAIL_SUBJECT Function
- FIND_IDENTIFIERS Function
- FIND_LINKS Function
- FIND_PHRASES Function
- FIND_TAGS Function
- GET_DOMAIN Function
- GET_FILE_EXTENSION Function
- GET_SLUG Function
- PHRASE_EXISTS Function
- REPLACE_WHITESPACE Function
- TO_DISPLAY_FILESIZE Function

56.1 DIFF Function

This function computes the difference between tables of lines. The implementation uses the default version of the longest common subexpression algorithm, without any optimizations. The DIFF function is not intended for very large inputs. The output is similar to the unified `diff` format.

Syntax

```
APEX_STRING_UTIL.FUNCTION DIFF (
    p_left    IN apex_t_varchar2,
    p_right   IN apex_t_varchar2,
    p_context IN PLS_INTEGER DEFAULT 3 )
RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_left	The lines in the "left" table.
p_right	The lines in the "right" table.
p_context	The number of same lines after each diff to also return (default 3).

Returns

A table of varchar2, where the first character denotes the type of diff:

- @ - Line numbers on left and right hand side.
- " " (space) - Context, left and right hand side are equal.
- -- Line is in left hand side, but not in right hand side.
- + - Line is in right hand side, but not in left hand side.

Example

This example computes the diff between the given tables.

```
select apex_string_util.diff (
    p_left  => apex_t_varchar2('how','now','brown','cow'),
    p_right => apex_t_varchar2('what','now','brown','cow',1,2,3) )
from sys.dual;

-> apex_t_varchar2 (
    '@@ 1,0 @@',
    '-how',
    '@@ 1,1 @@',
    '+what',
    ' now',
    ' brown',
    ' cow',
    '@@ 4,5 @@',
    '+1',
    '+2',
    '+3' )
```

56.2 FIND_EMAIL_ADDRESSES Function

This function finds all email addresses in the given input string.

Syntax

```
APEX_STRING_UTIL.FIND_EMAIL_ADDRESSES (
    p_string IN VARCHAR2 )
    RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_string	The input string.

Returns

This function returns an array of email addresses without duplicates.

Example

```

declare
    l_string varchar2(32767) := 'b@c.it hello this hello.world@example.com
is text b@c.it includes the ' ||
                                'michael.h@example.com email address and
x.y.z@m.io';
    l_results apex_t_varchar2;
begin
    l_results := apex_string_util.find_email_addresses(l_string);
end;
/
-> apex_t_varchar2 (
    'b@c.it',
    'hello.world@example.com',
    'michael.h@example.com',
    'x.y.z@m.io' )

```

56.3 FIND_EMAIL_FROM Function

Finds first occurrence of "From:" and the first email after the "From:".

Syntax

```
APEX_STRING_UTIL.FIND_EMAIL_FROM (
    p_string IN VARCHAR2 )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	The input string.

Returns

This function returns the from address.

Example

```

declare
    l_string varchar2(32767) := 'From: Marc Sample
<marc.sample@example.com>' || chr(10) ||
                                'Subject: Status Meeting'||chr(10)|||
                                'Date';
    l_result varchar2(4000);
begin
    l_result := apex_string_util.find_email_from(l_string);
    dbms_output.put_line('from = "'||l_result||'"');
end;
/
declare
    l_string varchar2(32767) := 'Elmar J. Fud <elmar.fud@example.com> wrote:';
```

```

    l_result varchar2(4000);
begin
    l_result := apex_string_util.find_email_from(l_string);
    dbms_output.put_line('from = ''||l_result||'''');
end;
/
-> from = "marc.sample@example.com"

```

56.4 FIND_EMAIL SUBJECT Function

This function finds the subject text in a given email string.

Syntax

```
APEX_STRING_UTIL.FIND_EMAIL_SUBJECT (
    p_string IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	The input string.

Returns

This function returns the subject line.

Example

```

declare
    l_string varchar2(32767) := 'From: Marc Sample
<marc.sample@example.com>'||chr(10) ||
                                'Subject: Status Meeting'||chr(10)||'
                                'Date';
    l_result varchar2(4000);
begin
    l_result := apex_string_util.find_email_subject(l_string);
    dbms_output.put_line('Subject = ''||l_result||'''');
end;
/
-> Subject = "Status meeting"

```

56.5 FIND_IDENTIFIERS Function

Given an identifier's prefix, this function finds the identifiers including consecutive numbers following. The search is case insensitive and also ignores white space and special characters.

Syntax

```
FUNCTION FIND_IDENTIFIERS (
    p_string IN VARCHAR2,
```

```
p_prefix IN VARCHAR2 )
RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_string	The input string.
p_prefix	The identifier prefix.

Returns

Returns an array of identifiers present in a string.

Example

```
DECLARE
    l_string  varchar2(32767) :=
        'ORA-02291: integrity constraint (A.B.C) violated - parent key not found
'|||
        'SR # 3-17627996921 bug: 23423 feature 100022 and feature: 1000001
rptno=28487031 sr# 1111111, ||
        ' i have filed bug 27911887.';
    l_results apex_t_varchar2;
BEGIN
    l_results := apex_string_util.find_identifiers(l_string,'ORA-');
    l_results := apex_string_util.find_identifiers(l_string,'sr ');
    l_results := apex_string_util.find_identifiers(l_string,'feature ');
    l_results := apex_string_util.find_identifiers(l_string,'bug ');
    l_results := apex_string_util.find_identifiers(l_string,'rptno=');
END;
/
-> apex_t_varchar2('ORA-02291')
-> apex_t_varchar2('SR 3-17627996921','SR 1111111')
-> apex_t_varchar2('FEATURE 100022','FEATURE 1000001')
-> apex_t_varchar2('BUG 23423','BUG 27911887')
-> apex_t_varchar2('RPTNO=28487031')
```

56.6 FIND_LINKS Function

This function finds `https` and `http` hypertext links within text. The URL is preserved and the protocol is returned in lower case.

Syntax

```
APEX_STRING_UTIL.FIND_LINKS (
    p_string      IN VARCHAR2,
    p_https_only IN BOOLEAN  DEFAULT FALSE )
RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_string	The input string.
p_https_only	Default FALSE.
	If TRUE, only returns https:// links.

Returns

This function returns an array of links.

Example

```
DECLARE
    l_string  varchar2(32767) := 'http://example.com i website.com like
https://carbuzz.com' ||
                                'and <a href="https://dpreview.com"> and
http://google.com';
    l_results apex_t_varchar2;
BEGIN
    l_results := apex_string_util.find_links(l_string,false);
END;
/
-> apex_t_string (
    'https://carbuzz.com',
    'https://dpreview.com',
    'http://google.com' )
```

56.7 FIND_PHRASES Function

This function finds the occurrences of p_string in p_phrase return in an array. The search is case insensitive and also ignores white space and special characters.

Syntax

```
APEX_STRING_UTIL.FIND_PHRASES (
    p_phrases IN apex_t_varchar2,
    p_string   IN VARCHAR2 )
RETURN apex_t_varchar2;
```

Parameters

Parameter	Description
p_phrases	A table of phrases.
p_string	The input string.

Returns

This function returns an array of phrases that were found, without duplicates.

Example

```

DECLARE
    l_phrases apex_t_varchar2 := apex_t_varchar2();
    l_arr     apex_t_varchar2 := apex_t_varchar2();
    l_string  varchar2(4000) := 'how now brown cow';
BEGIN
    apex_string.push(l_phrases,'brown');
    apex_string.push(l_phrases,'cow');
    apex_string.push(l_phrases,'brown cow');
    l_arr := apex_string_util.find_phrases(l_phrases,l_string);
END;
/
apex_t_varchar2('brown','cow','brown cow')

```

56.8 FIND_TAGS Function

This function finds all strings identified by a tag prefix. The search is case insensitive and also ignores white space and special characters.

This function searches for a tag prefix (such as #) at the start of a string or within the text after a space. This function also recognizes repeated tag prefixes (such as ##).

The return excludes the prefix identifier (tag instead of #tag).

Syntax

```

APEX_STRING_UTIL.FIND_TAGS (
    p_string          IN VARCHAR2,
    p_prefix          IN VARCHAR2 DEFAULT '#',
    p_exclude_numeric IN BOOLEAN DEFAULT TRUE )
RETURN apex_t_varchar2;

```

Parameters

Parameter	Description
p_string	The input string.
p_prefix	The tag prefix (default #).
p_exclude_numeric	If TRUE (default), excludes values that only contain the tag prefix and digits.

Returns

This function returns the found tags in upper case.

Example

```

DECLARE
    l_tags    apex_t_varchar2;
    l_string  varchar2(4000) := 'how now #orclapex @mike brown #cow';
BEGIN
    l_tags := apex_string_util.find_tags(l_string,'#');

```

```
    l_tags := apex_string_util.find_tags(l_string,'@');
END;
/
-> apex_t_varchar2('#ORCLAPEX','#COW')
-> apex_t_varchar2('@MIKE')
```

56.9 GET_DOMAIN Function

This function extracts a domain from a link or email.

Syntax

```
APEX_STRING_UTIL.GET_DOMAIN (
    p_string IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	The input string.

Returns

This function returns a domain from a url or email.

Example

```
select apex_string_util.get_domain('https://apex.oracle.com/en/platform/low-
code/') from dual
-> apex.oracle.com
```

56.10 GET_FILE_EXTENSION Function

This function returns a file name's extension.

Syntax

```
FUNCTION GET_FILE_EXTENSION (
    p_filename      IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_filename	The filename.

Returns

This function returns the file name's extension in lower case.

Example

The following example shows how to use the GET_FILE_EXTENSION function.

```
select apex_string_util.get_file_extension('foo.pPtx') from dual
-> ppx
select apex_string_util.get_file_extension('PLEASE.READ.ME.TXT') from dual
-> txt
```

56.11 GET_SLUG Function

Use this function to convert the input string to a "-" separated string, with special characters removed. The returned string contains a maximum of 255 characters in total, including hash (if requested).

Syntax

```
APEX_STRING_UTIL.GET_SLUG (
    p_string          IN VARCHAR2,
    p_hash_length     IN PLS_INTEGER DEFAULT 0 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	The input string.
p_hash_length	If > 0 (default is 0), append random digits to make the result unique. The longest hash that may be returned is 38 digits.

Example

```
select apex_string_util.get_slug('hey now, brown cow! 1') from dual;
-> hey-now-brown-cow-1
--
select apex_string_util.get_slug('hey now, brown cow! 1',4) from dual;
-> hey-now-brown-cow-1-3486
```

56.12 PHRASE_EXISTS Function

This function returns whether the given phrase is in p_string. The search is case insensitive and also ignores white space and special characters.

Syntax

```
APEX_STRING_UTIL.PHRASE_EXISTS (
    p_phrase    IN VARCHAR2,
    p_string    IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_phrase	The given phrase.
p_string	The input string.

Returns

This function returns TRUE if the phrase was found. Otherwise, this function returns FALSE.

Example

The following example shows how to use the PHRASE_EXISTS function.

```
DECLARE
    l_phrase varchar2(4000) := 'sqldeveloper';
    l_string varchar2(4000) := 'how now brown cow sqldeveloper? sql
developer.';
BEGIN
    IF apex_string_util.phrase_exists(l_phrase,l_string) then
        dbms_output.put_line('found');
    ELSE
        dbms_output.put_line('NOT found');
    END IF;
END;
/
-> found
```

56.13 REPLACE_WHITESPACE Function

This function can be used to tokenize the input. It replaces white space and special characters with the given whitespace character. It also lower-cases the input. If p_original_find contains '.' or '#', these characters are also replaced by white space.

Syntax

```
APEX_STRING_UTIL.REPLACE_WHITESPACE (
    p_string          IN VARCHAR,
    p_original_find   IN VARCHAR2 DEFAULT NULL,
    p_whitespace_character IN VARCHAR2 DEFAULT '|')
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	The input string.
p_original_find	A set of characters that were already found in a preceding search operation.
p_whitespace_character	The separator character.

Returns

This function returns the input string in lower case with all special characters replaced.

Example

```
select apex_string_util.replace_whitespace('foo: Bar...Baz') from dual
-> |foo|bar|baz|
select apex_string_util.replace_whitespace('foo: Bar...Baz',null,'*') from
dual
-> *foo*bar*baz*
select apex_string_util.replace_whitespace('foo: Bar...Baz','.','*') from dual
-> *foo*bar...baz*
```

56.14 TO_DISPLAY_FILESIZE Function

This function returns a friendly file size, given a size in bytes (for example, 5.1MB or 6GB).

Syntax

```
APEX_STRING_UTIL.TO_DISPLAY_FILESIZE (
    p_size_in_bytes      IN NUMBER )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_size_in_bytes	The input size in bytes.

Returns

Returns the file size with a unit.

Example

```
select apex_string_util.to_display_filesize(1312312312) from dual;
-> 1.2GB
```

APEX_THEME

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](#)

57.1 CLEAR_ALL_USERS_STYLE Procedure

This procedure clears all theme style user preferences for an application and theme.

Syntax

```
APEX_THEME.CLEAR_ALL_USERS_STYLE (
    p_application_id    IN NUMBER          DEFAULT {current application id},
    p_theme_number      IN NUMBER          DEFAULT {current theme id}
);
```

Parameters

Parameter	Description
p_application_id	The application to clear all user theme style preferences for.
p_theme_number	The theme number to clear all theme style user preferences for.

Example

The following example clears the all theme style user preferences for theme 42 in application 100.

```
apex_theme.clear_all_users_style(
    p_application_id => 100,
    p_theme_number  => 42
);
```

57.2 CLEAR_USER_STYLE Procedure

This procedure clears the theme style user preference for user and application.

Syntax

```
APEX_THEME.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

Parameter	Description
p_theme_number	The theme number to clear the theme style user preference.

Example

The following example clears the theme style user preference for the `ADMIN` user in application 100 and theme 42.

```
apex_theme.clear_user_style(
    p_application_id => 100,
    p_user           => 'ADMIN',
    p_theme_number   => 42
);
```

57.3 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

```
APEX_THEME.DISABLE_USER_STYLE (
    p_application_id  IN NUMBER          DEFAULT {current application id},
    p_theme_number    IN NUMBER          DEFAULT {current theme number}
);
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_theme_number	Number of user interface's <i>Current Theme</i> .

The following example disable end user theme style selection for the Desktop user interface of application 100.

```
declare
  l_theme_id apex_themes.theme_number%type;
begin
  select theme_number into l_theme_id
    from apex_appl_user_interfaces
   where application_id = 100
     and display_name    = 'Desktop';

  apex_theme.disable_user_style(
    p_application_id => 100,
    p_theme_number   => l_theme_id
  );
end;
```

57.4 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

```
APEX_THEME.ENABLE_USER_STYLE (
  p_application_id  IN NUMBER          DEFAULT {current application id},
  p_theme_number    IN NUMBER          DEFAULT {current theme number}
);
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_theme_number	Number of User Interface's <i>Current Theme</i> .

The following example enable end user theme style selection for the Desktop user interface of application 100.

```
declare
  l_theme_id apex_themes.theme_number%type;
begin
  select theme_number into l_theme_id
    from apex_appl_user_interfaces
   where application_id = 100
     and display_name    = 'Desktop';

  apex_theme.enable_user_style(
    p_application_id => 100,
```

```

    p_theme_number => l_theme_id
);
end;

```

57.5 GET_USER_STYLE Function

This function returns the theme style user preference for the user and application. If no user preference is present, it returns NULL.

Syntax

```
APEX_THEME.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

Parameter	Description
p_application_id	The application to set the user style preference.
p_user	The user name to the user style preference.
p_theme_number	The theme number to set the session style.
RETURN	The theme style ID which is set as a user preference.

Example

The query returns the theme style user preference for the ADMIN user in application 100 and theme 42.

```
select apex_theme.get_user_style( 100, 'ADMIN', 42 ) from dual;
```

57.6 SET_CURRENT_STYLE Procedure

This procedure sets current theme style for the current application.

Syntax

```
APEX_THEME.SET_CURRENT_STYLE (
    p_theme_number  IN NUMBER,
    p_id           IN VARCHAR2 );
```

Parameters

Parameter	Description
p_theme_number	The theme number for which to set the default style.
p_style_id	The ID of the new default theme style.

Example

The following example gets available theme styles from **APEX Dictionary View** for the DESKTOP user interface.

```
select s.theme_style_id, t.theme_number
  from apex_application_theme_styles s,
apex_application_themes t
 where s.application_id = t.application_id
   and s.theme_number = t.theme_number
   and s.application_id = :app_id
   and t.ui_type_name = 'DESKTOP'
   and s.is_current = 'Yes'
```

The following example sets the current theme style to one of values returned by the above query.

```
apex_theme.set_current_style (
  p_theme_number => {query.theme_number},
  p_id => {query.theme_style_id}
);
```

 **See Also:**

[SET_CURRENT_THEME_STYLE Procedure \(Deprecated\)](#)

57.7 SET_SESSION_STYLE Procedure

This procedure sets the theme style dynamically for the current session. This is typically called after successful authentication.

Syntax

```
APEX_THEME.SET_SESSION_STYLE (
  p_theme_number  IN NUMBER    DEFAULT {current theme number},
  p_name          IN VARCHAR2 );
```

Parameters

Parameter	Description
p_theme_number	The theme number to set the session style for. Default is the current theme of the application.
p_name	The name of the theme style to be used in the session.

Example

The following example gets the current theme number from **APEX Dictionary View** for the DESKTOP user interface.

```
select t.theme_number
  from apex_application_themes t
 where t.application_id = :app_id
   and t.ui_type_name = 'DESKTOP'
```

The following example sets the session theme style for the current theme to Vita.

```
apex_theme.set_session_style (
    p_theme_number => {query.theme_number},
    p_name => 'Vita'
);
```

57.8 SET_SESSION_STYLE_CSS Procedure

This procedure sets the theme style CSS URLs dynamically for the current session. Theme style CSS URLs directly pass in; a persistent style definition is optional. This is typically called after successful authentication.

Syntax

```
APEX_THEME.SET_SESSION_STYLE_CSS (
    p_theme_number IN NUMBER    DEFAULT {current theme number},
    p_css_file_urls IN VARCHAR2 );
```

Parameters

Parameter	Description
p_theme_number	The theme number to set the session style.
p_css_urls	The URLs to CSS files with style directives.

Example

The following example gets available theme styles from **Oracle APEX Dictionary View** for the DESKTOP user interface.

```
select s.theme_style_id, t.theme_number
  from apex_application_theme_styles s,
apex_application_themes t
 where s.application_id = t.application_id
   and s.theme_number = t.theme_number
   and s.application_id = :app_id
   and t.ui_type_name = 'DESKTOP'
   and s.is_current = 'Yes'
```

The following example sets the current theme style to one of values returned by the above query.

```
apex_theme.set_session_style_css(
    p_theme_number => {query.theme_number},
    p_css_urls => {URLs to theme style CSS files}
);
```

57.9 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

```
APEX_THEME.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

Parameter	Description
p_application_id	The application to set the user style preference.
p_user	The user name to the user style preference.
p_theme_number	The theme number to set the user style preference.
p_id	The ID of the theme style to set as a user preference.

Example

The following example gets available theme styles from **Oracle APEX 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 IDs as user preference for ADMIN in application ID 100.

```
apex_theme.set_user_style (
    p_application_id => 100,
    p_user           => 'ADMIN',
    p_theme_number   => {query.theme_number},
```

```
p_id          => {query.theme_style_id}  
);
```

APEX_UI_DEFAULT_UPDATE

The `APEX_UI_DEFAULT_UPDATE` package provides procedures to access user interface defaults from within SQL Developer or SQLcl.

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](#)
- [UPD_TABLE Procedure](#)

 See Also:

[Managing User Interface Defaults in Oracle APEX SQL Workshop Guide](#)

58.1 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

Parameter	Description
p_column_name	Name of column to be created.
p_label	Used for item label and report column heading.
p_help_text	Used for help text for items and interactive report columns
p_format_mask	Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.
p_default_value	Used as the default value for items.
p_form_format_mask	If provided, used as the format mask for items, overriding any value for the general format mask.
p_form_display_width	Used as the width of any items using this Attribute Definition.
p_form_display_height	Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).
p_form_data_type	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
p_report_format_mask	If provided, used as the format mask for report columns, overriding any value for the general format mask.
p_report_col_alignment	Used as the alignment for report column data (for example, number are usually right justified). Valid values are LEFT, CENTER, and RIGHT.

Parameter	Description
p_syn_name1	Name of synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.
p_syn_name2	Name of second synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.
p_syn_name3	Name of third synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.

Example

The following example creates a new attribute to the UI Defaults Attribute Dictionary within the workspace associated with the current schema. It also creates a synonym for that attribute.

```
BEGIN
    apex_ui_default_update.add_ad_column (
        p_column_name      => 'CREATED_BY',
        p_label             => 'Created By',
        p_help_text         => 'User that created the record.',
        p_form_display_width => 30,
        p_form_data_type   => 'VARCHAR',
        p_report_col_alignment => 'LEFT',
        p_syn_name1         => 'CREATED_BY_USER' );
END;
```

58.2 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

Parameter	Description
p_column_name	Name of column with the Attribute Dictionary that the synonym is being created for.
p_syn_name	Name of synonym to be created.

Example

The following example add the synonym CREATED_BY_USER to the CREATED_BY attribute of the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.add_ad_synonym (
        p_column_name => 'CREATED_BY',
        p_syn_name     => 'CREATED_BY_USER' );
END;
```

58.3 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

Parameter	Description
p_column_name	Name of column to be deleted

Example

The following example deletes the attribute CREATED_BY from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_column (
        p_column_name => 'CREATED_BY' );
END;
```

58.4 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

Parameter	Description
p_syn_name	Name of synonym to be deleted

Example

The following example deletes the synonym CREATED_BY_USER from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_synonym (
        p_syn_name    => 'CREATED_BY_USER' );
END;
```

58.5 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

Parameter	Description
p_table_name	Name of table whose column's UI Defaults are to be deleted.
p_column_name	Name of column whose UI Defaults are to be deleted.

Example

The following example deletes the column CREATED_BY from the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.del_column (
        p_table_name  => 'EMP',
        p_column_name => 'CREATED_BY' );
END;
```

58.6 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

Parameter	Description
p_table_name	Name of table whose group UI Defaults are to be deleted.
p_group_name	Name of group whose UI Defaults are to be deleted.

Example

The following example deletes the group AUDIT_INFO from the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.del_group (
        p_table_name => 'EMP',
        p_group_name => 'AUDIT_INFO' );
END;
```

58.7 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

Parameter	Description
p_table_name	Table name.

Example

The following example removes the UI Defaults for the EMP table that are associated with the current schema.

```
begin
    apex_ui_default_update.del_table (
        p_table_name => 'EMP' );
end;
/
```

58.8 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

Parameter	Description
p_table_name	Table name.

Example

The following example synchronizes the UI Defaults for the `EMP` table that are associated with the current schema.

```
BEGIN
    apex_ui_default_update.synch_table (
        p_table_name => 'EMP' );
END;
```

58.9 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

Parameter	Description
p_column_name	Name of column to be updated
p_new_column_name	New name for column, if column is being renamed
p_label	Used for item label and report column heading
p_help_text	Used for help text for items and interactive report columns
p_format_mask	Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.
p_default_value	Used as the default value for items.
p_form_format_mask	If provided, used as the format mask for items, overriding any value for the general format mask.
p_form_display_width	Used as the width of any items using this Attribute Definition.
p_form_display_height	Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).
p_form_data_type	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
p_report_format_mask	If provided, used as the format mask for report columns, overriding any value for the general format mask.
p_report_col_alignment	Used as the alignment for report column data (for example, number are usually right justified). Valid values are LEFT, CENTER, and RIGHT.

 **Note:**

If p_label through p_report_col_alignment are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the CREATED_BY column in the UI Defaults Attribute Dictionary within the workspace associated with the current schema, setting the form_format_mask to null.

```
BEGIN
    apex_ui_default_update.upd_ad_column (
        p_column_name      => 'CREATED_BY',
        p_form_format_mask => 'null%');
END;
```

58.10 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

Parameter	Description
p_syn_name	Name of synonym to be updated.
p_new_syn_name	New name for synonym.

Example

The following example updates the `CREATED_BY_USER` synonym in the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.upd_ad_synonym (
        p_syn_name      => 'CREATED_BY_USER',
        p_new_syn_name  => 'USER_CREATED_BY');
END;
```

58.11 UPD_COLUMN Procedure

If the provided table and column exists within the user's schema's table based User Interface Defaults, the provided parameters are updated. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_COLUMN (
    p_table_name          IN VARCHAR2,
    p_column_name         IN VARCHAR2,
    p_group_id            IN VARCHAR2 DEFAULT NULL,
    p_label                IN VARCHAR2 DEFAULT NULL,
    p_help_text             IN VARCHAR2 DEFAULT NULL,
    p_display_in_form       IN VARCHAR2 DEFAULT NULL,
    p_display_seq_form      IN VARCHAR2 DEFAULT NULL,
    p_mask_form              IN VARCHAR2 DEFAULT NULL,
    p_default_value          IN VARCHAR2 DEFAULT NULL,
    p_required                IN VARCHAR2 DEFAULT NULL,
    p_display_width           IN VARCHAR2 DEFAULT NULL,
    p_max_width               IN VARCHAR2 DEFAULT NULL,
    p_height                  IN VARCHAR2 DEFAULT NULL,
    p_display_in_report        IN VARCHAR2 DEFAULT NULL,
    p_display_seq_report       IN VARCHAR2 DEFAULT NULL,
    p_mask_report              IN VARCHAR2 DEFAULT NULL,
    p_alignment                 IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_table_name	Name of table whose column's UI Defaults are being updated.
p_column_name	Name of column whose UI Defaults are being updated.
p_group_id	ID of group to be associated with the column.
p_label	When creating a form against this table or view, this is used as the label for the item if this column is included. When creating a report or tabular form, this is used as the column heading if this column is included.
p_help_text	When creating a form against this table or view, this becomes the help text for the resulting item.
p_display_in_form	When creating a form against this table or view, this determines whether this column is displayed in the resulting form page. Valid values are Y and N.
p_display_seq_form	When creating a form against this table or view, this determines the sequence in which the columns are displayed in the resulting form page.
p_mask_form	When creating a form against this table or view, this specifies the mask that is applied to the item, such as 999-99-9999. This is not used for character based items.
p_default_value	When creating a form against this table or view, this specifies the default value for the item resulting from this column.
p_required	When creating a form against this table or view, this specifies to generate a validation in which the resulting item must be NOT NULL. Valid values are Y and N.
p_display_width	When creating a form against this table or view, this specifies the display width of the item resulting from this column.
p_max_width	When creating a form against this table or view, this specifies the maximum string length that a user is allowed to enter in the item resulting from this column.
p_height	When creating a form against this table or view, this specifies the display height of the item resulting from this column.
p_display_in_report	When creating a report against this table or view, this determines whether this column is displayed in the resulting report. Valid values are Y and N.
p_display_seq_report	When creating a report against this table or view, this determines the sequence in which the columns are displayed in the resulting report.
p_mask_report	When creating a report against this table or view, this specifies the mask that is applied against the data, such as 999-99-9999. This is not used for character based items.
p_alignment	When creating a report against this table or view, this determines the alignment for the resulting report column. Valid values are L for Left, C for Center, and R for Right.

 **Note:**

If p_group_id through p_alignment are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the column DEPT_NO within the EMP table definition within the UI Defaults Table Dictionary within the current schema, setting the group_id to null.

```
BEGIN
    apex_ui_default_update.upd_column (
        p_table_name      => 'EMP',
        p_column_name     => 'DEPT_NO',
        p_group_id        => 'null%' );
END;
```

58.12 UPD_DISPLAY_IN_FORM Procedure

The UPD_DISPLAY_IN_FORM procedure sets the display in form user interface defaults. This user interface default is used by wizards when you select to create a form based upon the table. It controls whether the column is included by default or not.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM (
    p_table_name          IN VARCHAR2,
    p_column_name         IN VARCHAR2,
    p_display_in_form     IN VARCHAR2 );
```

Parameters

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_display_in_form	Determines whether to display in the form by default, valid values are Y and N.

Example

In the following example, when creating a Form against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

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

58.13 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

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_display_in_report	Determines whether to display in the report by default, valid values are Y and N.

Example

In the following example, when creating a Report against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

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

58.14 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

Parameter	Description
p_table_name	Table name.
p_form_region_title	Desired form region title.

Example

This example demonstrates how to set the Forms Region Title user interface default on the DEPT table.

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

58.15 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

Parameter	Description
p_table_name	Name of table whose group is being updated.
p_group_name	Group being updated.
p_new_group_name	New name for group, if group is being renamed.
p_description	Description of group.
p_display_sequence	Display sequence of group.

 **Note:**

If p_description or p_display_sequence are set to 'null%', the value is nullified. If no value is passed in, that column is not updated.

Example

The following example updates the description of the group AUDIT_INFO within the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.upd_group (
        p_table_name  => 'EMP',
        p_group_name  => 'AUDIT_INFO',
```

```

    p_description => 'Audit columns' );
END;

```

58.16 UPD_ITEM_DISPLAY_HEIGHT Procedure

Sets the item display height user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column. Display height controls if the item is a text box or a text area.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT (
    p_table_name          IN VARCHAR2,
    p_column_name         IN VARCHAR2,
    p_display_height      IN NUMBER );
```

Parameters

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_display_height	Display height of any items created based upon this column.

Example

The following example sets a default item height of 3 when creating an item on the DNAME column against the DEPT table.

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

58.17 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

Parameter	Description
p_table_name	Table name.

Parameter	Description
p_column_name	Column name.
p_display_width	Display width of any items created based upon this column.

Example

The following example sets a default item width of 5 when creating an item on the DEPTNO column against the DEPT table.

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

58.18 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

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_format_mask	Format mask to be associated with the column.

Example

In the following example, when creating a Form against the EMP table, the default item format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

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

58.19 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

Parameter	Description
<code>p_table_name</code>	Table name.
<code>p_column_name</code>	Column name.
<code>p_help_text</code>	Desired help text.

Example

This example demonstrates how to set the User Interface Item Help Text default for the DEPTNO column in the DEPT table.

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

58.20 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

Parameter	Description
<code>p_table_name</code>	Table name.
<code>p_column_name</code>	Column name.
<code>p_label</code>	Desired item label.

Example

This example demonstrates how to set the User Interface Item Label default for the DEPTNO column in the DEPT table.

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

58.21 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

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_report_alignment	Defines the alignment of the column in a report. Valid values are L (left), C (center) and R (right).

Example

In the following example, when creating a Report against the DEPT table, the default column alignment on the DEPTNO column is set to Right justified.

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

58.22 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

Parameter	Description
p_table_name	Table name.
p_column_name	Column name.
p_format_mask	Format mask to be associated with the column whenever it is included in a report.

Example

In the following example, when creating a Report against the EMP table, the default format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

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

58.23 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

Parameter	Description
p_table_name	Table name.
p_report_region_title	Desired report region title.

Example

This example demonstrates how to set the Reports Region Title user interface default on the DEPT table.

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

58.24 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

Parameter	Description
p_table_name	Name of table being updated.
p_form_region_title	Region title used for forms.
p_report_region_title	Region title used for reports and tabular forms.

Note:

If null% is passed in for p_form_region_title or p_report_region_title, the value is set to null. If no value is passed in, that column is not updated.

Example

The following example updates the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
begin
    apex_ui_default_update.upd_table (
        p_table_name      => 'EMP',
        p_form_region_title  => 'Employee Details',
        p_report_region_title => 'Employees' );
end;
/
```

APEX_UTIL

The APEX_UTIL package provides utilities you can use when programming in the Oracle APEX environment. You can use the APEX_UTIL package to get and set session state, to get files, to check authorizations for users, to reset different states for users, to get and purge cache information, and to get and set preferences for users.

- [BLOB_TO_CLOB Function](#)
- [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](#)
- [CLOB_TO_BLOB 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 \(Deprecated\)](#)
- [DELETE_FEEDBACK Procedure](#)
- [DELETE_FEEDBACK_ATTACHMENT 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
- FEEDBACK_ENABLED Function
- 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_APEX_OWNER Function
- GET_APPLICATION_STATUS Function (Deprecated)
- GET_ATTRIBUTE Function
- GET_AUTHENTICATION_RESULT Function
- GET_BLOB_FILE_SRC Function
- GET_BUILD_OPTION_STATUS Function Signature 1 (Deprecated)
- GET_BUILD_OPTION_STATUS Function Signature 2 (Deprecated)
- 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 (Deprecated)
- 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 Signature 1
 - GET_SINCE Function Signature 2
 - 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
 - INCREMENT_CALENDAR Procedure (Deprecated)
 - 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
 - PRN Procedure
 - 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 Signature 1
- REMOVE_USER Procedure Signature 2
- REPLY_TO_FEEDBACK Procedure
- RESET_AUTHORIZATIONS Procedure (Deprecated)
- RESET_PASSWORD Procedure
- RESET_PW Procedure
- SAVEKEY_NUM Function
- SAVEKEY_VC2 Function
- SET_APP_BUILD_STATUS Procedure (Deprecated)
- SET_APPLICATION_STATUS Procedure (Deprecated)
- SET_ATTRIBUTE Procedure
- SET_AUTHENTICATION_RESULT Procedure
- SET_BUILD_OPTION_STATUS Procedure (Deprecated)
- 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 (Deprecated)
- 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
- TABLE_TO_STRING Function (Deprecated)
- UNEXPIRE_END_USER_ACCOUNT Procedure
- UNEXPIRE_WORKSPACE_ACCOUNT Procedure
- UNLOCK_ACCOUNT Procedure
- URL_ENCODE Function (Deprecated)
- WORKSPACE_ACCOUNT_DAYS_LEFT Function

59.1 BLOB_TO_CLOB Function

This function converts a BLOB to a temporary CLOB.

Syntax

```
APEX_UTIL.BLOB_TO_CLOB (
    p_blob          IN BLOB,
    p_charset       IN VARCHAR2 DEFAULT NULL,
    --
    p_in_memory     IN VARCHAR2 DEFAULT 'Y',
    p_free_immediately IN VARCHAR2 DEFAULT 'Y' )
RETURN CLOB;
```

Parameters

Parameter	Description
p_blob	BLOB to be converted to a CLOB.
p_charset	Character set of the BLOB to be converted. If omitted, the database character set is assumed and no character set conversion happens.
p_in_memory	If Y is specified, create the temporary LOB in memory.
p_free_immediately	If Y is specified, clean up the temporary LOB after the top-level call.

Returns

Temporary CLOB containing the BLOB contents.

Example

The following example grabs website contents as BLOB and converts it to a CLOB.

```

DECLARE
    l_clob clob;
    l_blob blob;
BEGIN
    l_blob := apex_web_service.make_rest_request_b(
        p_url => 'https://www.example.com/',
        p_http_method => 'GET' );

    l_clob := apex_util.blob_to_clob(
        p_blob => l_blob );

    sys.dbms_output.put_line( 'The CLOB has ' ||
        sys.dbms_lob.getlength( l_clob ) || ' bytes.' );
        sys.dbms_output.put_line( '-----' );
        sys.dbms_output.put_line( sys.dbms_lob.substr( l_clob, 80, 1 ) );
END;

```

59.2 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

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

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_PAGE_CACHE function to retrieve the cache date and time for page 9 of the currently executing application. If page 9 has been cached, the cache date and time is output using the HTP package. The page could have been cached either by the user issuing the call, or for all users if the page was not to be cached by the user.

```

DECLARE
    l_cache_date DATE DEFAULT NULL;
BEGIN
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE (
        p_application => :APP_ID,

```

```

        p_page => 9);
IF l_cache_date IS NOT NULL THEN
    HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
END IF;
END;

```

59.3 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

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

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_REGION_CACHE function to retrieve the cache date and time for the region named Cached Region on page 13 of the currently executing application. If the region has been cached, the cache date and time is output using the HTP package. The region could have been cached either by the user issuing the call, or for all users if the page was not to be cached by user.

```

DECLARE
    l_cache_date DATE DEFAULT NULL;
BEGIN
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE(
        p_application => :APP_ID,
        p_page => 13,
        p_region_name => 'Cached Region');
    IF l_cache_date IS NOT NULL THEN
        HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
    END IF;
END;

```

59.4 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

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

Example

The following example demonstrates how to use the CACHE_PURGE_BY_APPLICATION procedure to purge all the cached pages and regions for the application currently executing.

```
BEGIN
    APEX_UTIL.CACHE_PURGE_BY_APPLICATION(p_application => :APP_ID);
END;
```

59.5 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

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The page number (ID).
p_user_name	The user associated with cached pages and regions.

Example

The following example demonstrates how to use the CACHE_PURGE_BY_PAGE procedure to purge the cache for page 9 of the application currently executing. Additionally, if the p_user_name parameter is supplied, this procedure would be further restricted by a specific users cache (only relevant if the cache is set to be by user).

```
BEGIN
    APEX_UTIL.CACHE_PURGE_BY_PAGE(
        p_application => :APP_ID,
        p_page        => 9);
END;
```

59.6 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

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

Example

The following example demonstrates how to use the CACHE_PURGE_STALE procedure to purge all the stale pages and regions in the application currently executing.

```
BEGIN
    APEX_UTIL.CACHE_PURGE_STALE(p_application => :APP_ID);
END;
```

59.7 CHANGE_CURRENT_USER_PW Procedure

This procedure changes the password of the currently authenticated user.

Note:

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.CHANGE_CURRENT_USER_PW (
    p_new_password    IN VARCHAR2 );
```

Parameters

Parameter	Description
p_new_password	The new password value in clear text.

Example

The following example demonstrates how to use the `CHANGE_CURRENT_USER_PW` procedure to change the password for the user who is currently authenticated.

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

 **See Also:**

[RESET_PW Procedure](#)

59.8 CHANGE_PASSWORD_ON_FIRST_USE Function

This 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.

 **Note:**

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE (
    p_user_name      IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
<code>p_user_name</code>	The user name of the user account.

Example

The following example demonstrates how to use the `CHANGE_PASSWORD_ON_FIRST_USE` function. Use this function to check if the password of an APEX user account (workspace administrator,

developer, or end user) in the current workspace must be changed by the user the first time it is used.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM apex_users) LOOP
        IF APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE(p_user_name =>
c1.user_name) THEN
            http.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](#)

59.9 CLOB_TO_BLOB Function

This function converts a CLOB to a temporary BLOB.

Syntax

```
APEX_UTIL.CLOB_TO_BLOB (
    p_clob          IN CLOB,
    p_charset       IN VARCHAR2 DEFAULT NULL,
    p_include_bom  IN VARCHAR2 DEFAULT 'N',
    --
    p_in_memory     IN VARCHAR2 DEFAULT 'Y',
    p_free_immediately IN VARCHAR2 DEFAULT 'Y' )
RETURN BLOB;
```

Parameters

Parameter	Description
p_clob	CLOB to convert to a BLOB.
p_charset	Character set to convert the BLOB to. If omitted, no character set conversion happens.
p_include_bom	Prepend the generated BLOB with a BOM.
p_in_memory	If Y is specified, create the temporary LOB in memory.
p_free_immediately	If Y is specified, clean up the temporary LOB after the top-level call.

Returns

Temporary BLOB containing the CLOB contents.

Example

The following example converts a CLOB to a BLOB, with and without charset conversion.

```
DECLARE
    l_clob clob;
    l_blob blob;
BEGIN
    l_clob := to_clob( 'This is some CLOB content with umlauts: ü,ä,ö.' );

    l_blob := apex_util.clob_to_blob(
        p_clob => l_clob,
        p_charset => 'AL32UTF8' );

    sys.dbms_output.put_line( 'The utf-8 BLOB has ' ||
        sys.dbms_lob.getlength( l_blob ) || ' bytes.' );

    l_blob := apex_util.clob_to_blob(
        p_clob => l_clob,
        p_charset => 'WE8ISO8859P1' );

    sys.dbms_output.put_line( 'The iso-8859-1 BLOB has ' ||
        sys.dbms_lob.getlength( l_blob ) || ' bytes.' );
END;
```

59.10 CLOSE_OPEN_DB_LINKS Procedure

This procedure closes all open database links for the current database session.

It is rare for this procedure to be called programmatically in an application. The primary purpose of this procedure is for the middleware technology in an Oracle APEX environment (such as Oracle REST Data Service) to be configured such that it closes all of the open database links in a session, either before a request is made to the APEX engine, or after a request to the APEX 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 (ORDS) closes any open database links both before the request is made to the APEX engine and after the request is complete.

```
<entry key="procedure.postProcess">apex_util.close_open_db_links</entry>
<entry key="procedure.preProcess">apex_util.close_open_db_links</entry>
```

59.11 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

Parameter	Description
p_app_id	The ID of the application for which session state is cleared for current session.

Example

The following example demonstrates how to use the CLEAR_APP_CACHE procedure to clear all the current sessions state for the application with an ID of 100.

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

59.12 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

Parameter	Description
p_page_id	The ID of the page in the current application for which session state is cleared for current session.

Example

The following example demonstrates how to use the CLEAR_PAGE_CACHE procedure to clear the current session state for the page with an ID of 10.

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

59.13 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;
```

59.14 COUNT_CLICK Procedure

This procedure counts clicks from an application built in App Builder to an external site. You can also use the shorthand version, procedure `z`, in place of `APEX_UTIL.COUNT_CLICK`.

Syntax

```
APEX_UTIL.COUNT_CLICK (
    p_url          IN VARCHAR2,
    p_cat          IN VARCHAR2,
    p_id           IN VARCHAR2 DEFAULT NULL,
    p_user         IN VARCHAR2 DEFAULT NULL,
    p_workspace    IN VARCHAR2 DEFAULT NULL,
    p_referrer_policy IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_url	The URL to which to redirect.
p_cat	A category to classify the click.
p_id	(Optional) Secondary ID to associate with the click.
p_user	(Optional) The application user ID.
p_workspace	(Optional) The workspace associated with the application.
p_referrer_policy	The referrer-policy HTTP response header.

Example

The following example demonstrates how to use the COUNT_CLICK procedure to log how many users click on the `http://example.com` link specified. Once this information is logged, you can view it by using the APEX_WORKSPACE_CLICKS view and in the reports on this view available to workspace and site administrators.

```
DECLARE
    l_url VARCHAR2(255);
    l_cat VARCHAR2(30);
    l_workspace_id VARCHAR2(30);
BEGIN
    l_url := 'http://yahoo.com';
    l_cat := 'yahoo';
    l_workspace_id :=
TO_CHAR(APEX_UTIL.FIND_SECURITY_GROUP_ID('MY_WORKSPACE'));

    HTP.P('<a href=APEX_UTIL.COUNT_CLICK?p_url=' || l_url || '&p_cat=' || 
l_cat || '&p_workspace=' || l_workspace_id || '>Click</a>');
END;
```

 **See Also:**

- [FIND_SECURITY_GROUP_ID Function](#)
- Deleting Click Counting Log Entries in *Oracle APEX Administration Guide*
- Managing Authorized URLs in *Oracle APEX Administration Guide*

59.15 CREATE_USER Procedure

This procedure creates a new account record in the native Oracle APEX user accounts repository.

Use this procedure to programmatically create user accounts for applications that utilize the APEX Accounts authentication scheme. To execute this procedure within the context of an APEX application, the current user must be an APEX workspace administrator and the application must permit modification of the workspace repository.

When creating workspace developer or workspace administrator users, you must also ensure that the user can authenticate to the development environment authentication scheme. The CREATE_USER procedure only creates the APEX repository user. For example, if using Database Accounts authentication, you must also run `CREATE USER nnn IDENTIFIED BY yyy.`

Syntax

```
APEX_UTIL.CREATE_USER (
    p_user_id                      IN NUMBER      DEFAULT NULL,
    p_user_name                     IN VARCHAR2,
    p_first_name                    IN VARCHAR2 DEFAULT NULL,
    p_last_name                     IN VARCHAR2 DEFAULT NULL,
```

```

    p_description          IN VARCHAR2 DEFAULT NULL,
    p_email_address       IN VARCHAR2 DEFAULT NULL,
    p_web_password        IN VARCHAR2,
    p_web_password_format IN VARCHAR2 DEFAULT 'CLEAR_TEXT',
    p_group_ids           IN VARCHAR2 DEFAULT NULL,
    p_developer_privs    IN VARCHAR2 DEFAULT NULL,
    p_default_schema      IN VARCHAR2 DEFAULT NULL,
    p_default_date_format IN VARCHAR2 DEFAULT NULL,
    p_allow_access_to_schemas IN VARCHAR2 DEFAULT NULL,
    p_account_expiry      IN DATE    DEFAULT TRUNC(SYSDATE),
    p_account_locked      IN VARCHAR2 DEFAULT 'N',
    p_failed_access_attempts IN NUMBER  DEFAULT 0,
    p_change_password_on_first_use IN VARCHAR2 DEFAULT 'Y',
    p_first_password_use_occurred IN VARCHAR2 DEFAULT 'N',
    p_attribute_01         IN VARCHAR2 DEFAULT NULL,
    p_attribute_02         IN VARCHAR2 DEFAULT NULL,
    p_attribute_03         IN VARCHAR2 DEFAULT NULL,
    p_attribute_04         IN VARCHAR2 DEFAULT NULL,
    p_attribute_05         IN VARCHAR2 DEFAULT NULL,
    p_attribute_06         IN VARCHAR2 DEFAULT NULL,
    p_attribute_07         IN VARCHAR2 DEFAULT NULL,
    p_attribute_08         IN VARCHAR2 DEFAULT NULL,
    p_attribute_09         IN VARCHAR2 DEFAULT NULL,
    p_attribute_10         IN VARCHAR2 DEFAULT NULL,
    p_allow_app_building_yn IN VARCHAR2 DEFAULT NULL,
    p_allow_sql_workshop_yn IN VARCHAR2 DEFAULT NULL,
    p_allow_websheet_dev_yn IN VARCHAR2 DEFAULT NULL,
    p_allow_team_development_yn IN VARCHAR2 DEFAULT NULL );

```

Parameters

Parameter	Description
p_user_id	Numeric primary key of user account.
p_user_name	Alphanumeric name used for login.
p_first_name	Informational.
p_last_name	Informational.
p_description	Informational.
p_email_address	Email address.
p_web_password	Password.
p_web_password_format	If the value you are passing for the p_web_password parameter is in clear text format then use CLEAR_TEXT, otherwise use HEX_ENCODED_DIGEST_V2.
p_group_ids	Colon separated list of numeric group IDs.

Parameter	Description
p_developer_privs	<p>Colon separated list of developer privileges. If p_developer_privs is not null, the user is given access to Team Development. If p_developer_privs contains ADMIN, the user is given App Builder and SQL Workshop access. If p_developer_privs does not contain ADMIN but contains EDIT, the user is given App Builder access. If p_developer_privs does not contain ADMIN but contains SQL, the user is given SQL Workshop access.</p> <p>The following are acceptable values for this parameter:</p> <p>NULL - To create an end user (a user who can only authenticate to developed applications).</p> <p>CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - To create a user with developer privileges with access to App Builder and SQL Workshop.</p> <p>ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - To create a user with full workspace administrator and developer privileges with access to App Builder, SQL Workshop and Team Development.</p>

 **Note:**

Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER, and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs; EDIT_USER uses p_developer_roles; and FETCH_USER uses p_developer_role.

p_default_schema	A database schema assigned to the user's workspace, used by default for browsing.
p_default_date_format	Oracle Date format for user. Currently only used in SQL Workshop.
p_allow_access_to_schemas	Colon-separated list of schemas assigned to the user's workspace to which the user is restricted (leave NULL for all).
p_account_expiry	The date the password was last updated, which defaults to today's date on creation.
p_account_locked	Y or N indicating if account is locked or unlocked.
p_failed_access_attempts	Number of consecutive login failures that have occurred, defaults to 0 on creation.
p_change_password_on_first_use	Y or N to indicate whether password must be changed on first use, defaults to Y on creation.

Parameter	Description
p_first_password_use_occurred	Y or N to indicate whether login has occurred since password change, defaults to N on creation.
p_attribute_01 ... p_attribute_10	Arbitrary text accessible with an API.
p_allow_app_building_yn	Y or N to indicate whether access to App Builder is enabled.
p_allow_sql_workshop_yn	Y or N to indicate whether access to SQL Workshop is enabled..
p_allow_websheet_dev_yn	Y or N to indicate whether access to Websheet development is enabled.
p_allow_team_development_yn	Y or N to indicate whether access to Team Development is enabled.

Example 1

The following example creates an End User called NEWUSER1 with a password of secret99. End Users 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)
- and 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@example.com',
        p_web_password       => 'password',
        p_developer_privs   =>
'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL',
        p_default_schema     => 'MY_SCHEMA',
        p_allow_access_to_schemas => 'MY_SCHEMA2',
```

```

    p_change_password_on_first_use => 'N',
    p_attribute_01                  => '123 456 7890');
END;

```

 **See Also:**

- [FETCH_USER Procedure Signature 3](#)
- [EDIT_USER Procedure](#)
- [GET_GROUP_ID Function](#)

59.16 CREATE_USER_GROUP Procedure

This procedure creates a user group when you are using Oracle APEX Accounts authentication.

To execute this procedure within the context of an APEX application, the current user must be an APEX workspace administrator and the application must permit modification of the workspace repository.

 **Note:**

This procedure operates on the native APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.CREATE_USER_GROUP (
    p_id          IN NUMBER    DEFAULT NULL,
    p_group_name  IN VARCHAR2,
    p_security_group_id IN NUMBER    DEFAULT NULL,
    p_group_desc   IN VARCHAR2 DEFAULT NULL );
```

Parameter

Parameter	Description
p_id	Primary key of group.
p_group_name	Name of group.
p_security_group_id	Workspace ID.
p_group_desc	Descriptive text.

Example

The following example demonstrates how to use the CREATE_USER_GROUP procedure to create a new group called Managers with a description of text. Pass NULL for the p_id parameter to

enable 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;
```

59.17 CURRENT_USER_IN_GROUP Function

This function returns a Boolean result based on whether the current user is a member of the specified workspace group. You can use the group name or group ID to identify the group.

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

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

Parameter	Description
p_group_name	Identifies the name of an existing group in the workspace.
p_group_id	Identifies the numeric ID of an existing group in the workspace.

Example

The following example demonstrates how to use the CURRENT_USER_IN_GROUP function to check if the user currently authenticated belongs to the group Managers.

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

59.18 CUSTOM_CALENDAR Procedure (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use this procedure to change the existing calendar view to Custom Calendar.

Syntax

```
APEX_UTIL.CUSTOM_CALENDAR(  
    p_date_type_field IN VARCHAR2 );
```

Parameters

Parameter	Description
p_date_type_field	Identifies the item name used to define the type of calendar to be displayed.

Example 1

The following example defines a custom calendar based on the hidden calendar type field. Assuming the Calendar is created in Page 9, the following example hides the column called P9_CALENDAR_TYPE.

```
APEX_UTIL.CUSTOM_CALENDAR(  
    'P9_CALENDAR_TYPE');
```

59.19 DELETE_FEEDBACK Procedure

This procedure deletes feedback with the specified ID.

Syntax

```
APEX_UTIL.DELETE_FEEDBACK (  
    p_feedback_id      IN NUMBER )
```

Parameters

Parameter	Description
p_feedback_id	The ID of the feedback.

Example

The following example deletes a feedback with the specified ID.

```
BEGIN
    apex_util.delete_feedback (
        p_feedback_id      => 12345 );
END;
```

59.20 DELETE_FEEDBACK_ATTACHMENT Procedure

This procedure deletes the attachment of a feedback with the specified ID.

Syntax

```
APEX_UTIL.DELETE_FEEDBACK_ATTACHMENT (
    p_feedback_id      IN NUMBER )
```

Parameters

Parameter	Description
p_feedback_id	The ID of the feedback.

Example

The following example deletes the attachment of a feedback with the specified ID.

```
BEGIN
    apex_util.delete_feedback_attachment (
        p_feedback_id      => 12345 );
END;
```

59.21 DELETE_USER_GROUP Procedure Signature 1

This procedure deletes a user group by providing the primary key of the group when you are using Oracle APEX Accounts authentication. To execute this procedure, the current user must have administrative privileges in the workspace.

 **Note:**

This procedure operates on the native APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.DELETE_USER_GROUP (
    p_group_id    IN NUMBER );
```

Parameter

Parameter	Description
p_group_id	Primary key of group.

Example

The following example removes 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;
```

59.22 DELETE_USER_GROUP Procedure Signature 2

This procedure deletes a user group by providing the name of the group when you are using Oracle APEX Accounts authentication. To execute this procedure, the current user must have administrative privileges in the workspace.

 **Note:**

This procedure operates on the native APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.DELETE_USER_GROUP (
    p_group_name      IN VARCHAR2 );
```

Parameter

Parameter	Description
p_group_name	Name of group.

Example

The following example removes the user group Managers by providing the name of the user group.

```
BEGIN
    APEX_UTIL.DELETE_USER_GROUP (
```

```

    p_group_name => 'Managers');
END;

```

59.23 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 DEFAULT 'attachment',
    p_report_data        IN BLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type IN VARCHAR2 DEFAULT 'xsl-fo',
    p_document_format    IN VARCHAR2 DEFAULT 'pdf',
    p_print_server        IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline").
p_report_data	XML based report data.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

See Also:

Printing Report Regions in *Oracle APEX App Builder User's Guide*

59.24 DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2

This procedure initiates the download of a print document using a pre-defined report query and an RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name          IN VARCHAR,
    p_content_disposition IN VARCHAR DEFAULT 'attachment',
    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

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline").
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's Shared Components).
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 2 (Pre-defined report query and RTF or XSL-FO based report layout.). In this example, the data for the report is taken from a Report Query called 'ReportQueryAndXSL' stored in the current application's Shared Components > Report Queries. The report layout is taken from a value stored in a page item (P1_XSL).

```

BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name          => 'mydocument',
        p_content_disposition => 'attachment',
        p_application_id     => :APP_ID,
        p_report_query_name  => 'ReportQueryAndXSL',
        p_report_layout       => :P1_XSL,
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');
END;

```

See Also:

Printing Report Regions in *Oracle APEX App Builder User's Guide*

59.25 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 DEFAULT 'attachment',
    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

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline").
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's Shared Components).
p_report_layout_name	Name of the report layout (stored under application's Shared Components).
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 3 (Pre-defined report query and pre-defined report layout). In this example, the data for the report is taken from a Report Query called 'ReportQuery' stored in the current application's Shared Components > Report Queries. The report layout is taken from a Report Layout called 'ReportLayout' stored in the current application's Shared Components > Report Layouts. Note that if you want to provision dynamic layouts, instead of specifying 'ReportLayout' for the p_report_layout_name parameter, you could reference a page item that allowed the user to select one of multiple saved Report Layouts. This example also provides a way for the user to specify how they want to receive the document (as an attachment or inline), through passing the value of P1_CONTENT_DISP to the p_content_disposition parameter. P1_CONTENT_DISP is a page item of type 'Select List' with the following List of Values Definition:

```
STATIC2:In Browser:inline,Save / Open in separate Window;attachment
```

```
BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name          => 'myreport123',
        p_content_disposition => :P1_CONTENT_DISP,
        p_application_id     => :APP_ID,
        p_report_query_name  => 'ReportQuery',
        p_report_layout_name => 'ReportLayout',
        p_report_layout_type => 'rtf',
```

```

    p_document_format      => 'pdf');
END;

```

 **See Also:**

Printing Report Regions in *Oracle APEX App Builder User's Guide*

59.26 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

Parameter	Description
p_file_name	Defines the filename of the print document.
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline").
p_report_data	XML based report data, must be encoded in UTF-8.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 4 (XML based report data (as a CLOB) and RTF or XSL-FO based report layout). In this example both the report data (XML) and report layout (XSL-FO) are taken from values stored in page items.

```

BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name      => 'mydocument',
        p_content_disposition => 'attachment',
        p_report_data    => :P1_XML,
        p_report_layout   => :P1_XSL,
```

```

    p_report_layout_type  => 'xsl-fo',
    p_document_format      => 'pdf');
END;

```

 See Also:

[Printing Report Regions in Oracle APEX App Builder User's Guide](#)

59.27 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

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_user_name	Alphanumeric name used for login. See also SET_USERNAME Procedure
p_first_name	Informational. See also SET_FIRST_NAME Procedure
p_last_name	Informational. See also SET_LAST_NAME Procedure
p_web_password	Clear text password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.
p_new_password	Clear text new password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.
p_email_address	Informational. See also SET_EMAIL Procedure
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which the user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing.
p_group_ids	Colon-separated list of numeric group IDs.
p_developer_roles	Colon-separated list of developer privileges. The following are acceptable values for this parameter: <ul style="list-style-type: none"> • 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.

Parameter	Description
p_account_expiry	Date password was last updated. See also: <ul style="list-style-type: none">• EXPIRE_END_USER_ACCOUNT Procedure• EXPIRE_WORKSPACE_ACCOUNT Procedure• UNEXPIRE_END_USER_ACCOUNT Procedure• UNEXPIRE_WORKSPACE_ACCOUNT Procedure
p_account_locked	'Y' or 'N' indicating if account is locked or unlocked. See also: <ul style="list-style-type: none">• LOCK_ACCOUNT Procedure• UNLOCK_ACCOUNT Procedure
p_failed_access_attempts	Number of consecutive login failures that have occurred.
p_change_password_on_first_use	'Y' or 'N' to indicate whether password must be changed on first use. See also CHANGE_PASSWORD_ON_FIRST_USE Function
p_first_password_use_occurred	'Y' or 'N' to indicate whether login has occurred since password change. See also PASSWORD_FIRST_USE_OCCURRED Function

Example

The following example shows how to use the EDIT_USER procedure to update a user account. This example shows how you can use the EDIT_USER procedure to change the user 'FRANK' from a user with just developer privilege to a user with workspace administrator and developer privilege. Firstly, the FETCH_USER procedure is called to assign account details for the user 'FRANK' to local variables. These variables are then used in the call to EDIT_USER to preserve the details of the account, with the exception of the value for the p_developer_roles parameter, which is set to 'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL'.

```

DECLARE
    l_user_id                      NUMBER;
    l_workspace                     VARCHAR2(255);
    l_user_name                     VARCHAR2(100);
    l_first_name                    VARCHAR2(255);
    l_last_name                     VARCHAR2(255);
    l_web_password                 VARCHAR2(255);
    l_email_address                 VARCHAR2(240);
    l_start_date                    DATE;
    l_end_date                      DATE;
    l_employee_id                  NUMBER(15, 0);
    l_allow_access_to_schemas     VARCHAR2(4000);
    l_person_type                   VARCHAR2(1);
    l_default_schema                VARCHAR2(30);
    l_groups                        VARCHAR2(1000);
    l_developer_role                VARCHAR2(60);
    l_description                   VARCHAR2(240);
    l_account_expiry                DATE;
    l_account_locked                VARCHAR2(1);
    l_failed_access_attempts       NUMBER;
    l_change_password_on_first_use VARCHAR2(1);
    l_first_password_use_occurred  VARCHAR2(1);
BEGIN

```

```

l_user_id := APEX_UTIL.GET_USER_ID('FRANK');

APEX_UTIL.FETCH_USER(
    p_user_id                      => l_user_id,
    p_workspace                      => l_workspace,
    p_user_name                      => l_user_name,
    p_first_name                     => l_first_name,
    p_last_name                      => l_last_name,
    p_web_password                  => l_web_password,
    p_email_address                 => l_email_address,
    p_start_date                     => l_start_date,
    p_end_date                       => l_end_date,
    p_employee_id                   => l_employee_id,
    p_allow_access_to_schemas      => l_allow_access_to_schemas,
    p_person_type                    => l_person_type,
    p_default_schema                => l_default_schema,
    p_groups                         => l_groups,
    p_developer_role                => l_developer_role,
    p_description                    => l_description,
    p_account_expiry                => l_account_expiry,
    p_account_locked                => l_account_locked,
    p_failed_access_attempts        => l_failed_access_attempts,
    p_change_password_on_first_use => l_change_password_on_first_use,
    p_first_password_use_occurred   => l_first_password_use_occurred);

APEX_UTIL.EDIT_USER (
    p_user_id                      => l_user_id,
    p_user_name                      => l_user_name,
    p_first_name                     => l_first_name,
    p_last_name                      => l_last_name,
    p_web_password                  => l_web_password,
    p_new_password                  => l_web_password,
    p_email_address                 => l_email_address,
    p_start_date                     => l_start_date,
    p_end_date                       => l_end_date,
    p_employee_id                   => l_employee_id,
    p_allow_access_to_schemas      => l_allow_access_to_schemas,
    p_person_type                    => l_person_type,
    p_default_schema                => l_default_schema,
    p_group_ids                      => l_groups,
    p_developer_roles               =>
    'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL',
    p_description                    => l_description,
    p_account_expiry                => l_account_expiry,
    p_account_locked                => l_account_locked,
    p_failed_access_attempts        => l_failed_access_attempts,
    p_change_password_on_first_use => l_change_password_on_first_use,
    p_first_password_use_occurred   => l_first_password_use_occurred);

END;

```

 **See Also:**

[FETCH_USER Procedure Signature 3](#)

59.28 END_USER_ACCOUNT_DAYS_LEFT Function

This 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.

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT (
    p_user_name      IN VARCHAR2 );
RETURN NUMBER
```

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example determines the number of days remaining before an APEX 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](#)

59.29 EXPIRE_END_USER_ACCOUNT Procedure

This 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.

 **Note:**

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

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

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example expires an Oracle APEX 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 into 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](#)

59.30 EXPIRE_WORKSPACE_ACCOUNT Procedure

This 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

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the EXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to expire an Oracle APEX 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](#)

59.31 EXPORT_USERS Procedure

This procedure produces an export file of the current workspace definition, workspace users, and workspace groups when called from a page. 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

Parameter	Description
p_export_format	Indicates how rows in the export file are formatted. Specify UNIX to have the resulting file contain rows delimited by line feeds. Specify DOS to have the resulting file contain rows delimited by carriage returns and line feeds.

Example

The following example calls 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;
```

59.32 FEEDBACK_ENABLED Function

This function returns a boolean value to check if application feature Allow Feedback is enabled.

Syntax

```
APEX_UTIL.FEEDBACK_ENABLED
    RETURN boolean;
```

Parameters

None.

Example

The following example demonstrates how to use the FEEDBACK_ENABLED function. If Allow Feedback is enabled, TRUE is returned otherwise FALSE is returned.

```
BEGIN
    RETURN apex_util.feedback_enabled;
END;
```

59.33 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

Parameter	Description
p_item	The name of an application-level item (not a page item) whose current value is to be fetched.

Parameter	Description
p_app	The ID of the application that owns the item (leave null for the current application).
p_session	The session ID from which to obtain the value (leave null for the current session).

Example

The following example shows how to use the `FETCH_APP_ITEM` function to obtain the value of the application item '`F300_NAME`' in application 300. As no value is passed for `p_session`, this defaults to the current session state value.

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

59.34 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.

 **Note:**

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
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

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_workspace	The name of the workspace.
p_user_name	Alphanumeric name used for login. See also GET_USERNAME Function
p_first_name	Informational. See also GET_FIRST_NAME Function
p_last_name	Informational. See also GET_LAST_NAME Function
p_web_password	Obfuscated account password.
p_email_address	Email address. See also GET_EMAIL Function
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing. See also GET_DEFAULT_SCHEMA Function
p_groups	List of groups of which user is a member. See also <ul style="list-style-type: none">• GET_GROUPS_USER_BELONGS_TO Function• CURRENT_USER_IN_GROUP Function
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. See also GET_USER_ROLES Function
p_description	Informational.

Example

The following example shows how to use the `FETCH_USER` procedure with Signature 1. This procedure is passed the ID of the currently authenticated user for the only `IN` parameter `p_user_id`. The code then stores all the other `OUT` parameter values in local variables.

```
DECLARE
    l_workspace          VARCHAR2(255);
    l_user_name          VARCHAR2(100);
    l_first_name         VARCHAR2(255);
    l_last_name          VARCHAR2(255);
    l_web_password      VARCHAR2(255);
    l_email_address      VARCHAR2(240);
    l_start_date         DATE;
    l_end_date           DATE;
    l_employee_id        NUMBER(15,0);
    l_allow_access_to_schemas VARCHAR2(4000);
    l_person_type        VARCHAR2(1);
    l_default_schema     VARCHAR2(30);
    l_groups              VARCHAR2(1000);
    l_developer_role     VARCHAR2(60);
    l_description         VARCHAR2(240);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id            => APEX_UTIL.GET_CURRENT_USER_ID,
        p_workspace           => l_workspace,
        p_user_name           => l_user_name,
        p_first_name          => l_first_name,
        p_last_name           => l_last_name,
        p_web_password        => l_web_password,
        p_email_address       => l_email_address,
        p_start_date          => l_start_date,
        p_end_date             => l_end_date,
        p_employee_id         => l_employee_id,
        p_allow_access_to_schemas => l_allow_access_to_schemas,
        p_person_type          => l_person_type,
        p_default_schema       => l_default_schema,
        p_groups                => l_groups,
        p_developer_role       => l_developer_role,
        p_description           => l_description);
END;
```

See Also:

- [EDIT_USER Procedure](#)
- [GET_CURRENT_USER_ID Function](#)

59.35 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.

 **Note:**

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
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

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_user_name	Alphanumeric name used for login. See also GET_USERNAME Function
p_first_name	Informational. See also GET_FIRST_NAME Function
p_last_name	Informational. See also GET_LAST_NAME Function
p_email_address	Email address. See also GET_EMAIL Function
p_groups	List of groups of which user is a member. See also <ul style="list-style-type: none">• GET_GROUPS_USER_BELONGS_TO Function• CURRENT_USER_IN_GROUP Function

Parameter	Description
p_developer_role	<p>Colon-separated list of developer roles. The following are acceptable values for this parameter:</p> <p>null - Indicates an end user (a user who can only authenticate to developed applications).</p> <p>CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege.</p> <p>ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege.</p> <p>Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.</p> <p>See also GET_USER_ROLES Function</p>
p_description	Informational.

Example

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](#)

59.36 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.

 **Note:**

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.FETCH_USER (
    p_user_id                      IN          NUMBER,
    p_workspace                     OUT         VARCHAR2,
    p_user_name                     OUT         VARCHAR2,
    p_first_name                    OUT         VARCHAR2,
    p_last_name                     OUT         VARCHAR2,
    p_web_password                 OUT         VARCHAR2,
    p_email_address                OUT         VARCHAR2,
    p_start_date                   OUT         VARCHAR2,
    p_end_date                      OUT         VARCHAR2,
    p_employee_id                  OUT         VARCHAR2,
    p_allow_access_to_schemas     OUT         VARCHAR2,
    p_person_type                  OUT         VARCHAR2,
    p_default_schema               OUT         VARCHAR2,
    p_groups                        OUT         VARCHAR2,
    p_developer_role              OUT         VARCHAR2,
    p_description                  OUT         VARCHAR2,
    p_account_expiry               OUT         DATE,
    p_account_locked              OUT         VARCHAR2,
    p_failed_access_attempts      OUT         NUMBER,
    p_change_password_on_first_use OUT         VARCHAR2,
    p_first_password_use_occurred OUT         VARCHAR2 );
```

Parameters

Table 59-1 Fetch_User Parameters Signature 3

Parameter	Description
p_user_id	Numeric primary key of the user account.
p_workspace	The name of the workspace.
p_user_name	Alphanumeric name used for login. See also GET_USERNAME Function
p_first_name	Informational. See also GET_FIRST_NAME Function

Table 59-1 (Cont.) Fetch_User Parameters Signature 3

Parameter	Description
p_last_name	Informational. See also GET_LAST_NAME Function
p_web_password	Obfuscated account password.
p_email_address	Email address. See also GET_EMAIL Function
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing. See also GET_DEFAULT_SCHEMA Function
p_groups	List of groups of which user is a member. See also <ul style="list-style-type: none">• GET_GROUPS_USER_BELONGS_TO Function• CURRENT_USER_IN_GROUP Function
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. See also GET_USER_ROLES Function
p_description	Informational.
p_account_expiry	Date account password was last reset. See also <ul style="list-style-type: none">• END_USER_ACCOUNT_DAYS_LEFT Function• WORKSPACE_ACCOUNT_DAYS_LEFT Function
p_account_locked	Locked/Unlocked indicator Y or N. See also GET_ACCOUNT_LOCKED_STATUS Function
p_failed_access_attempts	Counter for consecutive login failures.
p_change_password_on_first_use	Setting to force password change on first use Y or N.
p_first_password_use_occurred	Indicates whether login with password occurred Y or N.

Example

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](#)

59.37 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

Parameter	Description
p_workspace	The name of the workspace.

Example

The following example demonstrates how to use the `FIND_SECURITY_GROUP_ID` function to return the security group ID for the workspace called 'DEMOS'.

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

59.38 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

Parameter	Description
p_security_group_id	The security group ID of a workspace.

Example

The following example demonstrates how to use the FIND_WORKSPACE function to return the workspace name for the workspace with a security group ID of 20.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.FIND_WORKSPACE (p_security_group_id =>'20');
END;
```

59.39 GET_ACCOUNT_LOCKED_STATUS Function

This 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.

 **Note:**

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS (
    p_user_name      IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example checks if an Oracle APEX 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](#)

59.40 GET_APEX_OWNER Function

This function returns the name of the schema containing the Oracle APEX engine.

Syntax

```
APEX_UTIL.GET_APEX_OWNER RETURN VARCHAR2;
```

Parameters

None.

Returns

The schema name which contains the APEX engine.

Example

```
DECLARE
    l_apex_owner varchar2(255);
BEGIN
    l_apex_owner := apex_util.get_apex_owner;
END;
```

59.41 GET_APPLICATION_STATUS Function (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use [GET_APPLICATION_STATUS Function](#) in APEX_APPLICATION_ADMIN instead.

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

Parameter	Description
p_application_id	The application ID.

Example

```
declare
    l_status varchar2(100);
begin
    l_status := apex_util.get_application_status(
        p_application_id => 117 );
    dbms_output.put_line( 'The current application status is: ' || l_status );
end;
```

See Also:

Availability in *Oracle APEX App Builder User's Guide*

59.42 GET_ATTRIBUTE Function

This function returns the value of one of the attribute values (1 through 10) of a named user in the Oracle APEX accounts table. 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

Parameter	Description
p_username	User name in the account.
p_attribute_number	Number of attributes in the user record (1 through 10).

Example

The following example returns 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](#)

59.43 GET_AUTHENTICATION_RESULT Function

Use this function to retrieve the authentication result of the current session. Any authenticated user can call this function in a page request context.

Syntax

```
APEX_UTIL.GET_AUTHENTICATION_RESULT  
RETURN NUMBER;
```

Parameters

None.

Example

The following example demonstrates how to use the post-authentication process of an application's authentication scheme to retrieve the authentication result code set during authentication.

```
APEX_UTIL.SET_SESSION_STATE('MY_AUTH_STATUS',  
    'Authentication result:'||APEX_UTIL.GET_AUTHENTICATION_RESULT);
```

 **See Also:**

- [SET_AUTHENTICATION_RESULT Procedure](#)
- [SET_CUSTOM_AUTH_STATUS Procedure](#)

59.44 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 more specific formatting of the display of the image (with height and width tags). This function must be called from a valid Oracle APEX session and also requires that the parameters that describe the BLOB are 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, the p_plain_url argument must be set to TRUE to ensure that no modal dialog code is 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

Parameter	Description
p_item_name	Name of valid application page item with type FILE that contains the source type of DB column.
p_v1	Value of primary key column 1.
p_v2	Value of primary key column 2.
p_content_disposition	Specify INLINE or ATTACHMENT, all other values ignored.

Example

As a PL/SQL Function Body:

```
RETURN '<img src="" || 
APEX_UTIL.GET_BLOB_FILE_SRC('P2_ATTACHMENT',:P2_EMPNO) || '" />';
```

As a Region Source of type SQL:

```
SELECT ID, NAME,CASE WHEN NVL(dbms_lob.getlength(document),0) = 0
THEN NULL
ELSE CASE WHEN attach_mimetype like 'image%'
THEN '<img src=""||apex_util.get_blob_file_src('P4_DOCUMENT',id)||'" />'||'
ELSE
'<a href=""||'
apex_util.get_blob_file_src('P4_DOCUMENT',id)||'">Download</a>'||'
end
END new_img
FROM TEST_WITH_BLOB
```

The previous example displays the BLOB within the report if it can be displayed, and provides a download link if it cannot be displayed.

See Also:

Understanding BLOB Support in Forms and Reports in *Oracle APEX App Builder User's Guide*

59.45 GET_BUILD_OPTION_STATUS Function Signature 1 (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use [GET_BUILD_OPTION_STATUS Function Signature 1](#) in APEX_APPLICATION_ADMIN instead.

Use this function to get the build option status of a specified application by providing the ID of the application build option.

Syntax

```
FUNCTION get_build_option_status(
    p_application_id    IN NUMBER,
    p_id                 IN NUMBER )
    RETURN varchar2;
```

Parameters

Parameters	Description
p_application_id	The ID of the application that owns the build option under shared components.
p_id	The ID of the build option in the application.

Example

The following code retrieves the current status of the specified build option that is identified by ID.

```
DECLARE
    l_status VARCHAR2(255);
BEGIN
    l_status := APEX_UTIL.GET_BUILD_OPTION_STATUS(
        P_APPLICATION_ID => 101,
        P_ID => 245935500311121039);
END;
/
```

59.46 GET_BUILD_OPTION_STATUS Function Signature 2 (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use [GET_BUILD_OPTION_STATUS Function Signature 2](#) in APEX_APPLICATION_ADMIN instead.

Use this function to get the build option status of a specified application by providing the name of the application build option.

Syntax

```
FUNCTION get_build_option_status (
    p_application_id      IN NUMBER
    p_build_option_name   IN VARCHAR2 )
    return VARCHAR2;
```

Parameters

Parameters	Description
p_application_id	The ID of the application that owns the build option under shared components.
p_build_option_name	The name of the build option in the application.

Example

The following code retrieves the current status of the specified build option that is identified by name.

```
DECLARE
    l_status VARCHAR2(255);
BEGIN
    l_status := APEX_UTIL.GET_BUILD_OPTION_STATUS(
        P_APPLICATION_ID => 101,
        P_BUILD_OPTION_NAME => 'EXCLUDE_FROM_PRODUCTION');
END;
/
```

59.47 GET_CURRENT_USER_ID Function

This function returns the numeric user ID of the current user.

 **Note:**

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

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;
```

59.48 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;
```

59.49 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;
```

59.50 GET_EMAIL Function

This function returns the email address associated with the named user.

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

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

Parameters

Parameter	Description
p_username	The user name in the account.

Example

The following example shows how to use the GET_EMAIL function to return the email address of the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(240);
BEGIN
    VAL := APEX_UTIL.GET_EMAIL(p_username => 'FRANK');
END;
```

 **See Also:**

[SET_EMAIL Procedure](#)

59.51 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

Parameter	Description
p_feedback_id	The unique identifier of the feedback item.
p_row	Identifies which follow-up to retrieve and is ordered by created_on_desc.
p_template	The template to use to return the follow up. Given the in the default template, the function can be used in a loop to return all the follow up to a feedback.

Example

The following example displays all the remaining follow-up for feedback with the ID of 123.

```
declare
    l_feedback_count  number;
begin
    select count(*)
    into l_feedback_count
    from apex_team_feedback_followup
    where feedback_id = 123;
```

```

for i in 1..1_feedback_count loop
    http.p(apex_util.get_feedback_follow_up (
        p_feedback_id => 123,
        p_row          => i,
        p_template     => '<br />#FOLLOW_UP# was created on
#CREATED_ON# by #CREATED_BY#') );
    end loop;
end;
/

```

59.52 GET_FILE Procedure

This procedure downloads files from the Oracle APEX file repository. If you invoke this procedure during page processing, ensure that no page branch is invoked under the same condition to avoid interference 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

Parameter	Description
p_file_id	ID in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace. The following example demonstrates how to use APEX_APPLICATION_FILES:
p_inline	Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment.

```

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;
```

Example

The following example returns 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](#)

59.53 GET_FILE_ID Function

This function obtains the primary key of a file in the Oracle APEX file repository.

Syntax

```
APEX_UTIL.GET_FILE_ID (
    p_name      IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_name	The NAME in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace.

Example

The following example retrieves 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;
```

59.54 GET_FIRST_NAME Function

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

 **Note:**

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

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

Parameters

Parameter	Description
p_username	Identifies the user name in the account.

Example

The following example shows how to use the GET_FIRST_NAME function to return the FIRST_NAME of the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_FIRST_NAME(p_username => 'FRANK');
END;
```

 **See Also:**

[SET_FIRST_NAME Procedure](#)

59.55 GET_GLOBAL_NOTIFICATION Function (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use [GET_GLOBAL_NOTIFICATION Function](#) in APEX_APPLICATION_ADMIN instead.

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

Parameter	Description
p_application_id	The application ID.

Example

```
declare
    l_global_notification varchar2(100);

begin
    l_global_notification := apex_util.get_global_notification(
        p_application_id => 117 );
    dbms_output.put_line( 'The current global notification is: ' ||
    l_global_notification );
end;
```

See Also:

Availability in *Oracle APEX App Builder User's Guide*

59.56 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.

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

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

Parameters

Parameter	Description
p_username	Identifies the user name in the account.

Example

The following example shows how to use the GET_GROUPS_USER_BELONGS_TO to return the list of groups to which the user 'FRANK' is a member.

```
DECLARE
    VAL VARCHAR2(32765);
BEGIN
    VAL := APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'FRANK');
END;
```

See Also:

[EDIT_USER Procedure](#)

59.57 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

Parameter	Description
p_group_name	Identifies the name of a group in the workspace.

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;
```

59.58 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

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

Example

The following example shows how to use the `GET_GROUP_NAME` function to return the name of the group with the ID 8922003.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;
```

59.59 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_salts IN BOOLEAN DEFAULT TRUE )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_values	The input values.
p_salts	If TRUE (default), salt hash with internal session information.

Example

This example updates the SAL and COMM columns of a given record in the EMP table, but throws an error if the column data has changed in the meantime.

```
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;
```

59.60 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

Parameter	Description
p_on_message	Optional text used for the link to switch to high contrast mode, when you are in standard mode. If this parameter is not passed, the default 'Set High Contrast Mode On' text is returned in the link.
p_off_message	Optional text used for the link to switch to standard mode, when you are in high contrast mode. If this parameter is not passed, the default 'Set High Contrast Mode Off' text is returned in the link.

Example

When running in standard mode, this function returns a link with the text 'Set High Contrast Mode On'. When the link is clicked the current page is refreshed and high contrast mode is switched on. When running in high contrast mode, a link 'Set High Contrast Mode Off' is returned. When the link is clicked the current page is refreshed and switched back to standard mode.

```
BEGIN
    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](#)

59.61 GET_LAST_NAME Function

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

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

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

Parameters

Parameter	Description
<code>p_username</code>	The user name in the user account record.

Example

The following example shows how to use the function to return the `LAST_NAME` for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_LAST_NAME(p_username => 'FRANK');
END;
```

 **See Also:**

[SET_LAST_NAME Procedure](#)

59.62 GET_NUMERIC_SESSION_STATE Function

This function returns a numeric value for a numeric item. You can use this function in Oracle APEX applications wherever you can use PL/SQL or SQL. You can also use the shorthand function `NV` in place of `APEX_UTIL.GET_NUMERIC_SESSION_STATE`.

 **Tip:**

For enhanced query performance, use FAST DUAL functionality in the following SQL code syntax:

```
(select apex_util.get_numeric_session_state('P1_ITEM') from dual)
```

Syntax

```
APEX_UTIL.GET_NUMERIC_SESSION_STATE (
    p_item  IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

Parameter	Description
<code>p_item</code>	The case insensitive name of the item for which you want to have the session state fetched.

Example

The following example shows how to use the function to return the numeric value stored in session state for the item `my_item`.

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

 **See Also:**

- [GET_SESSION_STATE Function](#)
- [SET_SESSION_STATE Procedure](#)

59.63 GET_PREFERENCE Function

This function retrieves the value of a previously saved preference for a given user.

Syntax

```
APEX_UTIL.GET_PREFERENCE (
    p_preference  IN      VARCHAR2 DEFAULT NULL,
    p_user        IN      VARCHAR2 DEFAULT V('USER') )
RETURN VARCHAR2;
```

Parameters

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

Example

The following example shows how to use the `GET_PREFERENCE` function to return the value for the currently authenticated user's preference named `default_view`.

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

 See Also:

- [SET_PREFERENCE Procedure](#)
- [REMOVE_PREFERENCE Procedure](#)
- Managing User Preferences in *Oracle APEX Administration Guide*

59.64 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

Parameter	Description
p_report_data	XML based report data.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

For a `GET_PRINT_DOCUMENT` example see [GET_PRINT_DOCUMENT Function Signature 4](#).

59.65 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

Parameter	Description
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's Shared Components).
p_report_layout_name	Name of the report layout (stored under application's Shared Components).
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

For a `GET_PRINT_DOCUMENT` example see [GET_PRINT_DOCUMENT Function Signature 4](#).

59.66 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

Parameter	Description
p_application_id	Defines the application ID of the report query.
p_report_query_name	Name of the report query (stored under application's shared components).
p_report_layout	Defines the report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

For a `GET_PRINT_DOCUMENT` example see [GET_PRINT_DOCUMENT Function Signature 4](#).

59.67 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

Parameter	Description
p_report_data	XML based report data, must be encoded in UTF-8.
p_report_layout	Report layout in XSL-FO or RTF format.
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf".
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml".
p_print_server	URL of the print server. If not specified, the print server is derived from preferences.

Example

The following example shows how to use the GET_PRINT_DOCUMENT using Signature 4 (Document returns as a BLOB using XML based report data and RTF or XSL-FO based report layout). In this example, GET_PRINT_DOCUMENT is used with APEX_MAIL.SEND and APEX_MAIL.ADD_ATTACHMENT to send an email with an attachment of the file returned by GET_PRINT_DOCUMENT. Both the report data and layout are taken from values stored in page items (P1_XML and P1_XSL).

```
DECLARE
    l_id number;
    l_document BLOB;
BEGIN
    l_document := APEX_UTIL.GET_PRINT_DOCUMENT (
        p_report_data      => :P1_XML,
        p_report_layout    => :P1_XSL,
        p_report_layout_type => 'xsl-fo',
        p_document_format  => 'pdf');

    l_id := APEX_MAIL.SEND(
        p_to      => :P35_MAIL_TO,
        p_from    => 'admin@example.com',
        p_subj    => 'sending PDF by using print API',
        p_body    => 'Please review the attachment.',
        p_body_html => 'Please review the attachment');

    APEX_MAIL.ADD_ATTACHMENT (
```

```

    p_mail_id      => l_id,
    p_attachment   => l_document,
    p_filename     => 'mydocument.pdf',
    p_mime_type    => 'application/pdf');
END;

```

59.68 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

Parameter	Description
p_on_message	Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text is returned in the link.
p_off_message	Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text is returned in the link.

Example

When running in standard mode, this function returns a link with the text 'Set Screen Reader Mode On'. When the link is clicked the current page is refreshed and screen reader mode is switched on. When running in screen reader mode, a link with text 'Set Screen Reader 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_screen_reader_mode_toggle);
END;

```

See Also:

[SHOW_SCREEN_READER_MODE_TOGGLE Procedure](#)

59.69 GET_SESSION_LANG Function

This function returns the language setting for the current user in the current Oracle APEX session.

Syntax

```
APEX_UTIL.GET_SESSION_LANG
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the session language for the current user in the current APEX session into a local variable.

```
DECLARE
    VAL VARCHAR2(5);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_LANG;
END;
```

59.70 GET_SESSION_STATE Function

This function returns the value for an item. You can use this function in your Oracle APEX applications wherever you can use PL/SQL or SQL. You can also use the shorthand function `▼` in place of `APEX_UTIL.GET_SESSION_STATE`.

Tip:

For enhanced query performance, use FAST DUAL functionality in the following SQL code syntax:

```
(select apex_util.get_session_state('P1_ITEM') from dual)
```

Syntax

```
APEX_UTIL.GET_SESSION_STATE (
    p_item IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_item	The case insensitive name of the item for which you want to have the session state fetched.

Example

The following example returns 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](#)

59.71 GET_SESSION_TERRITORY Function

This function returns the territory setting for the current user in the current Oracle APEX session.

Syntax

```
APEX_UTIL.GET_SESSION_TERRITORY
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example returns the session territory setting for the current user in the current APEX session into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_TERRITORY;
END;
```

59.72 GET_SESSION_TIME_ZONE Function

This function returns the time zone for the current user in the current Oracle APEX 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 returns the session time zone for the current user in the current APEX session into a local variable.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := apex_util.get_session_time_zone;
END;
```

59.73 GET_SINCE Function Signature 1

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" or "30m" and so forth. This function is equivalent to using the format masks `SINCE` and `SINCE_SHORT` available within Oracle APEX 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

Parameter	Description
<code>p_date</code>	The date you want formatted.
<code>p_short</code>	Boolean or Y/N to indicate whether to return a short version of relative date.

Example

```
select application_id, application_name, apex_util.get_since(last_updated_on)
last_update
    from apex_applications
order by application_id
```

59.74 GET_SINCE Function Signature 2

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" or "30m" and so forth. This function is equivalent to using the format masks `SINCE` and `SINCE_SHORT` available within Oracle APEX and is useful within SQL queries or PL/SQL routines.

Syntax

```
APEX_UTIL.GET_SINCE (
    p_value in [ timestamp | timestamp with time zone | timestamp with local
    time zone ],
    p_short in [ boolean default false | varchar2 default 'N' ] )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
<code>p_value</code>	The <code>TIMESTAMP</code> , <code>TIMESTAMP WITH TIME ZONE</code> , <code>TIMESTAMP WITH LOCAL TIME ZONE</code> you want to format.
<code>p_short</code>	Boolean or Y/N to indicate whether to return a short version of relative date.

Examples

This example 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 example 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
```

59.75 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

Parameter	Description
p_application_id	The application ID to get supporting objects from.
p_script_type	The supporting objects script type. Valid values are apex_util.c_install_script, apex_util.c_upgrade_script, apex_util.c_deinstall_script.

Example

The following example shows how to set workspace ID for workspace FRED, then get supporting objects from application ID 100.

```
declare
    l_install_script    clob;
    l_upgrade_script    clob;
    l_deinstall_script clob;
begin
    apex_util.set_workspace( p_workspace => 'FRED');

    l_install_script :=
apex_util.get_supporting_object_script( p_application_id => 100,
    p_script_type => apex_util.c_install_script );
    l_upgrade_script :=
apex_util.get_supporting_object_script( p_application_id => 100,
    p_script_type => apex_util.c_upgrade_script );
    l_deinstall_script :=
apex_util.get_supporting_object_script( p_application_id => 100,
    p_script_type => apex_util.c_deinstall_script );
end;
```

59.76 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

Parameter	Description
p_application_id	The application ID to get supporting objects from.
p_script_type	The supporting objects script type. Valid values are apex_util.c_install_script, apex_util.c_upgrade_script, apex_util.c_deinstall_script.
p_output_type	The script can be output to sys.dbms_output buffer or download as a file. Valid values are apex_util.c_output_as_dbms_output, apex_util.c_output_as_file. The default is c_output_as_dbms_output.

Example 1

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;
```

Example 2

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;
```

59.77 GET_USER_ID Function

This function returns the numeric ID of a named user in the workspace.

 **Note:**

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.GET_USER_ID (
    p_username    IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

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

Example

The following example shows how to use the `GET_USER_ID` function to return the ID for the user named 'FRANK'.

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

59.78 GET_USER_ROLES Function

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

Syntax

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

Parameters

Parameter	Description
p_username	Identifies a user name in the account.

Example

The following example returns a 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;
```

See Also:

- [CREATE_USER Procedure](#)
- [EDIT_USER Procedure](#)
- [FETCH_USER Procedure Signature 1](#)
- [FETCH_USER Procedure Signature 2](#)
- [FETCH_USER Procedure Signature 3](#)

59.79 GET_USERNAME Function

This function returns the user name of a user account identified by a numeric ID.

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with Oracle APEX Accounts authentication.

Syntax

```
APEX_UTIL.GET_USERNAME (
    p_userid    IN NUMBER )
RETURN VARCHAR2;
```

Parameters

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

Example

The following example uses `GET_USERNAME` to return the user name for the user with an ID of 228922003.

```
DECLARE
    val varchar2(100);
BEGIN
    val := apex_util.get_username(p_userid => 228922003);
END;
```

 See Also:[SET_USERNAME Procedure](#)

59.80 HOST_URL Function

This function returns the URL to the Oracle APEX instance, depending on the option passed.

Syntax

```
APEX_UTIL.HOST_URL (
    p_option    IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_option	<p>Specifies the parts of the URL to include.</p> <p>Possible values for p_option include:</p> <ul style="list-style-type: none"> NULL - Return URL up to port number. For example: <code>http://example.com:7778</code> SCRIPT - Return URL to include script name. For example: For example (Friendly URL enabled): <code>https://example.com:7778/pls/apex/r/{workspace}/application</code> For example (Friendly URL disabled) <code>https://example.com:7778/pls/apex/</code> APEX_PATH - Return URL to include the APEX path. For example: <code>https://example.com:7778/pls/apex/</code> IMGPRE - Return URL to include image prefix. For example: <code>https://example.com:7778/i/</code>

Example

The following example returns the URL to the current APEX instance including the script name.

```
declare
    l_host_url      varchar2(4000);
    l_url           varchar2(4000);
    l_application   varchar2(30) := 'f?p=100:1';
    l_email_body    varchar2(32000);
begin
    l_host_url := apex_util.host_url('SCRIPT');
    l_url := l_host_url||l_application;
    l_email_body := 'The URL to the application is: '||l_url;
end;
```

59.81 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,
    p_aria_labelledby IN VARCHAR2   DEFAULT NULL,
    p_aria_label      IN VARCHAR2   DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_number	Number between 0 and 100.
p_size	Width of graph in pixels.
p_background	Six character hexadecimal background color of chart bar (not bar color).
p_bar_background	Six character hexadecimal background color of chart bar (bar color).
p_format	If this parameter is supplied, p_size, p_background and p_bar_background are ignored. This parameter uses the following format: PCT_GRAPH:<BACKGROUND>:<FOREGROUND>:<CHART_WIDTH> position 1: PCT_GRAPH format mask indicator position 2: Background color in hexadecimal, 6 characters (optional) position 3: Foreground "bar" color in hexadecimal, 6 characters (optional) position 4: Chart width in pixels. Numeric and defaults to 100. p_number is automatically scaled so that 50 is half of chart_width (optional).
p_aria_labelledby	Space-separated list of one or more ID values referencing the elements that label the percent graph.
p_aria_label	Value that labels the percent graph.

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
```

59.82 INCREMENT_CALENDAR Procedure (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

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;
```

Parameters

None.

Example

In this example, if you create a button called NEXT in the Calendar page and a process that fires when the create button is clicked, the following code navigates the calendar.

```
APEX_UTIL.INCREMENT_CALENDAR
```

59.83 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. Only use this procedure in a page submit process.

Syntax

```
APEX_UTIL.IR_CLEAR (
    p_page_id      IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.

Parameter	Description
p_report_alias	Identifies the saved report alias within the current application page. To clear a Primary report, set p_report_alias to PRIMARY or leave as NULL. To clear a saved report, p_report_alias must be the name of the saved report. For example, to clear report 1234, set p_report_alias to 1234.

Example

The following example clears interactive report settings with alias of 8101021 in page 1 of the current application.

```
BEGIN
    APEX_UTIL.IR_CLEAR(
        p_page_id      => 1,
        p_report_alias => '8101021'
    );
END;
```

59.84 IR_DELETE_REPORT Procedure (Deprecated)

 **Note:**

Use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure deletes saved interactive reports. It deletes all saved reports except the Primary Default report.

Syntax

```
APEX_UTIL.IR_DELETE_REPORT (
    p_report_id      IN NUMBER );
```

Parameters

Parameter	Description
p_report_id	Report ID to delete within the current Oracle APEX application.

Example

The following example shows how to use the IR_DELETE_REPORT procedure to delete the saved Interactive report with ID of '880629800374638220' in the current application.

```
BEGIN
    APEX_UTIL.IR_DELETE_REPORT(
        p_report_id => '880629800374638220');
END;
```

59.85 IR_DELETE_SUBSCRIPTION Procedure (Deprecated)

 **Note:**

The use of this procedure is not recommended. This procedure has been replaced by the procedure in APEX_IR.

This procedure deletes Interactive subscriptions.

Syntax

```
APEX_UTIL.IR_DELETE_SUBSCRIPTION(
    p_subscription_id IN NUMBER);
```

Parameters

Parameter	Description
p_subscription_id	Subscription ID to delete within the current workspace.

Example

The following example shows how to use the IR_DELETE_SUBSCRIPTION procedure to delete the subscription with ID of '880629800374638220' in the current workspace.

```
BEGIN
    APEX_UTIL.IR_DELETE_SUBSCRIPTION(
        p_subscription_id => '880629800374638220');
END;
```

59.86 IR_FILTER Procedure (Deprecated)

 **Note:**

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. Only use this procedure 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_report_column	Name of the report SQL column, or column alias, to be filtered.
p_operator_abbr	Filter type. Valid values are as follows: <ul style="list-style-type: none"> • EQ = Equals • NEQ = Not Equals • LT = Less than • LTE = Less then or equal to • GT = Greater Than • GTE = Greater than or equal to • LIKE = SQL Like operator • N = Null • NN = Not Null • C = Contains • NC = Not Contains • IN = SQL In Operator • NIN = SQL Not In Operator
p_filter_value	Filter value. This value is not used for N and NN.
p_report_alias	Identifies the saved report alias within the current application page. To create a filter on a Primary report, p_report_alias must be PRIMARY or leave as NULL. To create a filter on a saved report, p_report_alias must be the name of the saved report. For example, to create a filter on report 1234, p_report_alias must be 1234.

Example

The following example shows how to use the IR_FILTER procedure to filter interactive report with alias of 8101021 in page 1 of the current application with DEPTNO equals 30.

```
BEGIN
    APEX_UTIL.IR_FILTER (
        p_page_id      => 1,
        p_report_column => 'DEPTNO',
        p_operator_abbr => 'EQ',
        p_filter_value   => '30'
        p_report_alias   => '8101021'
    );
END;
```

59.87 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

Parameter	Description
p_page_id	Page of the current Oracle APEX application that contains an interactive report.
p_report_alias	Identifies the saved report alias within the current application page. To reset a Primary report, p_report_alias must be 'PRIMARY' or leave as NULL. To reset a saved report, p_report_alias must be the name of the saved report. For example, to reset report '1234', p_report_alias must be '1234'.

Example

The following example shows how to use the IR_RESET procedure to reset Interactive report settings with alias of '8101021' in page 1 of the current application.

```
BEGIN  
    APEX_UTIL.IR_RESET(  
        p_page_id      => 1,  
        p_report_alias => '8101021'  
    );  
END;
```

59.88 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;
```

59.89 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;
```

59.90 IS_LOGIN_PASSWORD_VALID Function

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with APEX Accounts authentication.

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

Parameter	Description
p_username	User name in account.
p_password	Password to be compared with password stored in the account.

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.

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

59.91 IS_SCREEN_READER_SESSION Function

This function returns a boolean TRUE if the session is in screen reader mode and returns a boolean FALSE if not in screen reader mode.

Syntax

```
APEX_UTIL.IS_SCREEN_READER_SESSION
RETURN BOOLEAN;
```

Parameters

None

Example

```
BEGIN
    IF apex_util.is_screen_reader_session then
        htp.p('Screen Reader Mode');
    END IF;
END;
```

59.92 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;
```

59.93 IS_USERNAME_UNIQUE Function

Note:

This function operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with APEX Accounts authentication.

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

Parameter	Description
p_username	Identifies the user name to be tested.

Example

The following example shows how to use the IS_USERNAME_UNIQUE function. If the user 'FRANK' already exists in the current workspace, FALSE is returned, otherwise TRUE is returned.

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

59.94 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](#)

59.95 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](#)

59.96 LOCK_ACCOUNT Procedure

This 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

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example locks all Oracle APEX accounts (workspace administrator, developer, or end user) in the current workspace.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_workspace_apex_users ) LOOP
        APEX_UTIL.LOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('User Account:'||c1.user_name||' is now locked.');
    END LOOP;
END;
```

See Also:

- [UNLOCK_ACCOUNT Procedure](#)
- [GET_ACCOUNT_LOCKED_STATUS Function](#)

59.97 PASSWORD_FIRST_USE_OCCURRED Function

This function returns `TRUE` if the account's password has changed since the account was created, an Oracle APEX 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

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example demonstrates how to check if the password for an APEX 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
        http.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](#)

Note:

Oracle recommends using `APEX_PAGE.GET_URL` instead of `PREPARE_URL` for improved readability.

See [GET_URL Function](#).

The `PREPARE_URL` function serves two purposes:

1. To return an APEX navigation URL with the Session State Protection checksum argument (&cs=) if one is required. For security, the URL will not contain a checksum if the specified application is located in a different workspace.
2. To return an APEX navigation 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 function returns the APEX navigation URL with &cs=<large hex value> appended. If you use this returned value (such as in JavaScript), you may need 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

Parameter	Description
p_url	An APEX navigation URL with all substitutions resolved.
p_url_charset	The character set name (for example, UTF-8) to use when escaping special characters contained within argument values.
p_checksum_type	Null or any of the following values: <ul style="list-style-type: none"> • PUBLIC_BOOKMARK or 1 - Use this when generating links to be used by any user. For example, use this value when generating an email which includes links to an application. • PRIVATE_BOOKMARK or 2 - Use this when generating a link to be used outside of the current session. This option can only be used by the same currently authenticated user. • SESSION or 3 - Use this when generating links to an application. This option can only be used within the current session.
p_triggering_element	A jQuery selector (for example, #my_button , where my_button is the static ID for a button element), to identify which element to use to trigger the dialog. This is required for Modal Dialog support.
p_plain_url	If the page you are calling APEX_UTIL.PREPARE_URL from is a modal dialog, specify p_plain_url to omit the unnecessary dialog code in the generated link. By default, if this function is called from a modal dialog, extra code to close the modal dialog is included in the generated URL.

Example 1

The following example shows how to use the `PREPARE_URL` function to return a URL with a valid 'SESSION' level checksum argument. This URL sets the value of `P1_ITEM` page item to `xyz`.

```
DECLARE
    l_url varchar2(2000);
    l_app number := v('APP_ID');
    l_session number := v('APP_SESSION');
BEGIN
    l_url := APEX_UTIL.PREPARE_URL(
        p_url => 'f?p=' || l_app || ':1:' || l_session || '::NO:::P1_ITEM:xyz',
        p_checksum_type => 'SESSION');
END;
```

Example 2

The following example shows how to use the `PREPARE_URL` function to return a URL with a zero session ID. In a PL/SQL Dynamic Content region that generates `f?p` URLs (anchors), call `PREPARE_URL` to ensure that the session ID is set to zero when the zero session ID feature is in use, when the user is a public user (not authenticated), and when the target page is a public page in the current application:

```
htp.p(APEX_UTIL.PREPARE_URL(p_url => 'f?p=' || :APP_ID ||
':10:' || :APP_SESSION
|| '::NO:::P10_ITEM:ABC'));
```

When using `PREPARE_URL` for this purpose, the `p_url_charset` and `p_checksum_type` arguments can be omitted. However, it is permissible to use them when both the Session State Protection and Zero Session ID features are applicable.

 **See Also:**

About Enabling Support for Bookmarks in *Oracle APEX App Builder User's Guide*

59.99 PRN Procedure

This procedure prints a given CLOB to the HTP buffer.

Syntax

```
APEX_UTIL.PRN (
    p_clob    IN CLOB,
    p_escape  IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
p_clob	The CLOB.
p_escape	If TRUE (default), escape special characters, using apex_escape.html.

Example

The following example prints l_clob and escape special characters.

```
DECLARE
    l_clob clob := '<script>alert(1)</script>';
BEGIN
    apex_util.prn (
        p_clob    => l_clob,
        p_escape => true );
END;
```

59.100 PUBLIC_CHECK_AUTHORIZATION Function (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use the [IS_AUTHORIZED Function](#) instead of this deprecated function.

Given the name of a authorization scheme, this function determines if the current user passes the security check.

Syntax

```
APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION (
    p_security_scheme    IN      VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_security_scheme	The name of the authorization scheme that determines if the user passes the security check.

Example

The following example shows how to use the PUBLIC_CHECK_AUTHORIZATION function to check if the current user passes the check defined in the `my_auth_scheme` authorization scheme.

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

59.101 PURGE_REGIONS_BY_APP Procedure

Deletes all cached regions for an application.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_APP (
    p_application IN NUMBER );
```

Parameters

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

Example

The following example shows 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;
```

59.102 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

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

Example

The following example shows how to use the PURGE_REGIONS_BY_NAME procedure to delete all the cached values for the region 'my_cached_region' on page 1 of the current application.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_NAME(
        p_application => :APP_ID,
        p_page => 1,
        p_region_name => 'my_cached_region');
END;
```

59.103 PURGE_REGIONS_BY_PAGE Procedure

Deletes all cached regions by application and page.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_PAGE (
    p_application IN NUMBER,
    p_page         IN NUMBER );
```

Parameters

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

Example

The following example shows how to use the PURGE_REGIONS_BY_PAGE procedure to delete all the cached values for regions on page 1 of the current application.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_PAGE(
        p_application => :APP_ID,
        p_page => 1);
END;
```

59.104 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 cancel further processing of the Oracle APEX application.

Syntax

```
APEX_UTIL.REDIRECT_URL (
    p_url          IN VARCHAR2,
    p_reset_htp_buffer IN BOOLEAN DEFAULT TRUE );
```

Parameters

Parameter	Description
<code>p_url</code>	The URL the browser requests.
<code>p_reset_htp_buffer</code>	Set to <code>TRUE</code> 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 <code>FALSE</code> if the application has its own cookie to use in the response.

Example

The following example tells the browser to redirect to `http://www.example.com` and immediately stops further processing.

```
apex_util.redirect_url (
    p_url => 'http://www.example.com/' );
```

59.105 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

Parameter	Description
<code>p_preference</code>	Name of the preference to remove.
<code>p_user</code>	User for whom the preference is defined.

Example

The following example shows how to use the REMOVE_PREFERENCE procedure to remove the preference default_view for the currently authenticated user.

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

See Also:

- [GET_PREFERENCE Function](#)
- [SET_PREFERENCE Procedure](#)
- Managing User Preferences in *Oracle APEX Administration Guide*

59.106 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

Parameter	Description
p_user	Identifies the user for whom sorting preferences are removed.

Example

The following example shows how to use the REMOVE_SORT_PREFERENCES procedure to remove the currently authenticated user's column heading sorting preferences.

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

59.107 REMOVE_USER Procedure Signature 1

This procedure removes the user account identified by the primary key. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.REMOVE_USER (
    p_user_id    IN NUMBER );
```

Parameters

Parameter	Description
p_user_id	The numeric primary key of the user account record.

Example

The following examples show how to use the REMOVE_USER procedure to remove a user account by the primary key using the p_user_id parameter.

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

59.108 REMOVE_USER Procedure Signature 2

This procedure removes the user account identified by the user name. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

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

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following examples show how to use the REMOVE_USER procedure to remove a user account by user name using the p_user_name parameter.

```
BEGIN
    FOR i in 1..10 LOOP
        APEX_UTIL.REMOVE_USER(
            p_user_name => 'USER_'||i);
    END LOOP;
    COMMIT;
END;
```

59.109 REPLY_TO_FEEDBACK Procedure

This procedure enables you to submit a reply to a feedback.

Syntax

```
APEX_UTIL.REPLY_TO_FEEDBACK (
    p_feedback_id      IN NUMBER,
    p_type             IN NUMBER  DEFAULT NULL,
    p_status            IN NUMBER  DEFAULT NULL,
    p_tags              IN VARCHAR2 DEFAULT NULL,
    p_developer_comment IN VARCHAR2 DEFAULT NULL,
    p_public_response   IN VARCHAR2 DEFAULT NULL,
    p_followup          IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_feedback_id	The ID of feedback.
p_type	The type of feedback (1 is General Comment, 2 is Enhancement Request, 3 is Bug).
p_status	The status of the feedback (0 is No Action, 1 is Acknowledged, 3 is Open and 4 is Closed).
p_tags	Used to categorize feedback, only to be displayed to developers (not in the current feature implementation).
p_developer_comment	Comments not displayed to the users but could be displayed to developers (not in the current feature implementation).
p_public_response	Text that will be displayed to the users.
p_followup	The text of the follow-up. When the Feedback feature is added to an application, it only uses p_public_response. It does not support passing data to p_followup.

Example 1

In this example, the app admin replies to a feedback submitted by a user reporting a bug.

```
BEGIN
    apex_util.reply_to_feedback (
        p_feedback_id      => 12345,
        p_type             => 3,
        p_status            => 1,
        p_developer_comment => 'Created JIRA BUG-123!',
        p_public_response   => 'Thank you for letting us know about
this! We will fix it in the upcoming release!',
        p_followup          => 'We really appreciate your support of our
application.' );
END;
```

Example 2

In this example, the developer leaves a comment about creating a ticket for the bug reported by the user, and closes the feedback workflow.

```
BEGIN
    reply_to_feedback (
        p_feedback_id      => 12345,
        p_type             => 3,
        p_status           => 4,
        p_developer_comment => 'Filed ticket BUG-123!',
        p_followup         => 'We really appreciate your support of our
application.' );
END;
```

59.110 RESET_AUTHORIZATIONS Procedure (Deprecated)



Note:

Use the [RESET_CACHE Procedure](#) instead of this deprecated procedure.

To increase performance, Oracle APEX 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;
```

59.111 RESET_PASSWORD Procedure

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
apex_application.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

Parameter	Description
p_user_name	The user whose password should be changed. The default is the currently logged in Oracle APEX user name.
p_old_password	The current password of the user. The call succeeds if the given value matches the current password or it is NULL and the owner of the calling PL/SQL code has APEX_ADMINISTRATOR_ROLE. If the value is not the user's password, an error occurs.
p_new_password	The new password.
p_change_password_on_first_use	If TRUE (default), the user must change the password on the next login.

Errors Raised

Error	Description
INVALID_CREDENTIALS	Occurs if p_user_name does not match p_old_password.
APEX.AUTHENTICATION.LOGIN_THROTTLE.COUNTER	Indicates authentication prevented by login throttle.
internal_error	Occurs if p_old_password is NULL and caller does not have APEX_ADMINISTRATOR_ROLE.
internal_error	Indicates caller is not a valid workspace schema.

Example

This example demonstrates changing the password of the currently logged-in user to a new password.

```
apex_util.reset_password (
    p_old_password => :P111_OLD_PASSWORD,
    p_new_password => :P111_NEW_PASSWORD );
```

59.112 RESET_PW Procedure

This procedure resets the password for a named user and emails it in a message to the email address located for the named account in the current workspace. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.RESET_PW (
    p_user IN VARCHAR2,
    p_msg   IN VARCHAR2 );
```

Parameters

Parameter	Description
p_user	The user name of the user account.
p_msg	Message text to be mailed to a user.

Example

The following example shows how to use the `RESET_PW` procedure to reset the password for the user 'FRANK'.

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

See Also:

[CHANGE_CURRENT_USER_PW Procedure](#)

59.113 SAVEKEY_NUM Function

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

Syntax

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

Parameters

Parameter	Description
p_val	The numeric value to be saved.

Example

The following example shows how to use the SAVEKEY_NUM function to set the apex_utilities.g_val_num package variable to the value of 10.

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

 **See Also:**

[KEYVAL_NUM Function](#)

59.114 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

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

Example

The following example shows how to use the `SAVEKEY_VC2` function to set the `apex_utilities.g_val_vc2` package variable to the value of 'XXX'.

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

 **See Also:**

[KEYVAL_VC2 Function](#)

59.115 SET_APP_BUILD_STATUS Procedure (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use [SET_BUILD_STATUS Procedure](#) in APEX_APPLICATION_ADMIN instead.

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

Parameter	Description
p_application_id	The ID of the application.
p_build_status	The new build status of the application. Values include: <ul style="list-style-type: none">RUN_ONLY - The application can be run but cannot be edited by developers.RUN_AND_BUILD - The application can be run and can also be edited by developers.

Example

```
begin
    apex_util.set_app_build_status(
        p_application_id => 170,
        p_build_status    => 'RUN_ONLY' );
    commit;
end;
```

59.116 SET_APPLICATION_STATUS Procedure (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use [SET_APPLICATION_STATUS Procedure](#) in APEX_APPLICATION_ADMIN instead.

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

Parameter	Description
p_application_id	The Application ID.
p_application_status	New application status. Values include: <ul style="list-style-type: none">• AVAILABLE - Application is available with no restrictions.• AVAILABLE_W_EDIT_LINK - Application is available with no restrictions. Developer Toolbar shown to developers.• DEVELOPERS_ONLY - Application only available to developers.• RESTRICTED_ACCESS - Application only available to users in p_restricted_user_list.• UNAVAILABLE - Application unavailable. Message shown in p_unavailable_value.• UNAVAILABLE_PLSQL - Application unavailable. Message shown from PL/SQL block in p_unavailable_value.• UNAVAILABLE_URL - Application unavailable. Redirected to URL provided in p_unavailable_value.
p_unavailable_value	Value used when application is unavailable. This value has different semantics dependent upon value for p_application_status.
p_restricted_user_list	Comma separated list of users permitted to access application, when p_application_status = RESTRICTED_ACCESS.

Examples

```
begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'AVAILABLE' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'AVAILABLE_W_EDIT_LINK' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'DEVELOPERS_ONLY' );
end;
```

```
begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'RESTRICTED_ACCESS',
    p_restricted_user_list => 'xxx.xxx@example.com' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE',
    p_unavailable_value => 'Application not available, sorry' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE_PLSQL',
    p_unavailable_value => 'sys.htp.p('''Application unavailable,
sorry''');' );
end;

begin
apex_util.set_application_status(
    p_application_id => 117,
    p_application_status => 'UNAVAILABLE_URL',
    p_unavailable_value => 'http://www.example.com' );
end;
```

 See Also:

Availability in *Oracle APEX App Builder User's Guide*

59.117 SET_ATTRIBUTE Procedure

This procedure sets the value of one of the attribute values (1 through 10) of a user in the Oracle APEX accounts table.

Syntax

```
APEX_UTIL.SET_ATTRIBUTE (
    p_userid          IN NUMBER,
    p_attribute_number IN NUMBER,
    p_attribute_value  IN VARCHAR2 );
```

Parameters

Parameter	Description
p_userid	The numeric ID of the user account.

Parameter	Description
p_attribute_number	Attribute number in the user record (1 through 10).
p_attribute_value	Value of the attribute located by p_attribute_number to be set in the user record.

Example

The following example sets the number 1 attribute for user FRANK with the value foo.

```
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](#)

59.118 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

Parameter	Description
p_code	Any numeric value the developer chooses. After this value is set in the session using this procedure, it can be retrieved using the APEX_UTIL.GET_AUTHENTICATION_RESULT function.

Example

One way to use this procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. In this example, no credentials verification is performed, it just demonstrates how text and numeric status values can be registered for logging. Note that the status set using this

procedure is visible in the `apex_user_access_log` view and in the reports on this view available to workspace and site administrators.

```
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is
back.');
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

See Also:

- [GET_AUTHENTICATION_RESULT Function](#)
- [SET_CUSTOM_AUTH_STATUS Procedure](#)
- Monitoring Activity within a Workspace in *Oracle APEX Administration Guide*

59.119 SET_BUILD_OPTION_STATUS Procedure (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use [SET_BUILD_OPTION_STATUS Procedure](#) in `APEX_APPLICATION_ADMIN` instead.

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

Parameter	Description
p_application_id	The ID of the application that owns the build option under shared components.
p_id	The ID of the build option in the application.
p_build_status	The new status of the build option. Possible values are INCLUDE, EXCLUDE both upper case.

Example

The following example demonstrates how to use the SET_BUILD_OPTION_STATUS procedure to change the current status of build option.

```
BEGIN
APEX_UTIL.SET_BUILD_OPTION_STATUS(
    P_APPLICATION_ID => 101,
    P_ID => 245935500311121039, P_BUILD_STATUS=>'INCLUDE');

END;
```

59.120 SET_CURRENT_THEME_STYLE Procedure (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

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

Parameter	Description
p_theme_number	The current theme number of the application. This can be retrieved from APEX_APPLICATION_THEMES view.
p_theme_style_id	The numeric ID of theme style. You can get available theme styles for an application from APEX_APPLICATION_THEME_STYLES view.

Example

The following example shows how to use the SET_CURRENT_THEME_STYLE procedure to set the current application desktop theme style to Blue.

```

DECLARE
    l_current_theme_number number;
    l_theme_style_id       number;

BEGIN
    select theme_number
    into l_current_theme_number
    from apex_application_themes
    where application_id = :app_id
    and ui_type_name      = 'DESKTOP'
    and is_current = 'Yes';

    select s.theme_style_id
    into l_new_theme_style_id
    from apex_application_theme_styles s, apex_application_themes t
    where s.application_id = t.application_id
    and s.theme_number = t.theme_number
    and s.application_id = :app_id
    and t.ui_type_name     = 'DESKTOP'
    and t.is_current = 'Yes'
    and s.name = 'Blue';

    IF l_current_theme_number IS NOT NULL AND
    l_new_theme_style_id IS NOT NULL THEN
        APEX_UTIL.SET_CURRENT_THEME_STYLE(
            p_theme_number    => l_current_theme_number,
            p_theme_style_id => l_new_theme_style_id
        );
    END IF;

END;

```

See Also:

[SET_CURRENT_STYLE Procedure](#)

59.121 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

Parameter	Description
p_status	Any text the developer chooses to denote the result of the authentication attempt (up to 4000 characters).

Example

One way to use the `SET_CUSTOM_AUTH_STATUS` procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. Note that no credentials verification is performed. The status set using this procedure is visible in the `apex_user_access_log` view and in the reports on this view available to workspace and site administrators.

```
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is
back.');
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

See Also:

- [SET_AUTHENTICATION_RESULT Procedure](#)
- [GET_AUTHENTICATION_RESULT Function](#)
- Monitoring Activity within a Workspace in *Oracle APEX Administration Guide*

59.122 SET_EDITION Procedure

This procedure sets the name of the edition to be used in all application SQL parsed in the current page view or page submission.

Syntax

```
APEX_UTIL.SET_EDITION (
    p_edition IN VARCHAR2 );
```

Parameters

Parameter	Description
p_edition	Edition name.

Example

The following example shows how to use the SET_EDITION procedure. It sets the edition name for the database session of the current page view.

```
BEGIN
    APEX_UTIL.SET_EDITION( P_EDITION => 'Edition1' );
END;
```

Note:

Support for Edition-based Redefinition requires Oracle Database version 11.2.0.1 or later.

59.123 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

SET_EMAIL Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_email	The email address to be saved in user account.

Example

The following example shows how to use the SET_EMAIL procedure to set the value of EMAIL to "frank.scott@example.com" for the user "FRANK."

```
BEGIN
    APEX_UTIL.SET_EMAIL(
        p_userid  => APEX_UTIL.GET_USER_ID('FRANK'),
        p_email    => 'frank.scott@example.com');
END;
```

See Also:

- [GET_EMAIL Function](#)
- [GET_USER_ID Function](#)

59.124 SET_FIRST_NAME Procedure

Note:

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with APEX Accounts authentication.

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

Parameter	Description
p_userid	The numeric ID of the user account.
p_first_name	FIRST_NAME value to be saved in user account.

Example

The following example shows how to use the SET_FIRST_NAME procedure to set the value of FIRST_NAME to 'FRANK' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_FIRST_NAME(
        p_userid      => APEX_UTIL.GET_USER_ID('FRANK'),
        p_first_name  => 'FRANK');
END;
```

See Also:

- [GET_FIRST_NAME Function](#)
- [GET_USER_ID Function](#)

59.125 SET_GLOBAL_NOTIFICATION Procedure (Deprecated)

Note:

This API is deprecated and will be removed in a future release.

Use [SET_GLOBAL_NOTIFICATION Procedure](#) in APEX_APPLICATION_ADMIN instead.

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

Parameter	Description
p_application_id	The Application ID.
p_global_notification_message	Text string to be used for the global notification message.

Example

```
BEGIN
    apex_util.set_global_notification(
        p_application_id      => 117,
```

```

    p_global_notification_message => 'This application will be upgraded
this weekend at 2100 UTC' );
END;

```

 **See Also:**

Availability in *Oracle APEX App Builder User's Guide*

59.126 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

Parameter	Description
p_group_name	The target group name.
p_granted_group_names	The names of groups to grant to p_group_name.

Example

This example creates three groups (ACCTS_PAY, ACCTS_REC, MANAGER) and then grants ACCTS_PAY and ACCTS_REC to MANAGER.

```

apex_util.create_user_group (
    p_group_name => 'ACCTS_PAY' );
apex_util.create_user_group (
    p_group_name => 'ACCTS_REC' );
apex_util.create_user_group (
    p_group_name => 'MANAGER' );
apex_util.set_group_group_grants (
    p_group_name => 'MANAGER',
    p_granted_group_names => apex_t_varchar2('ACCTS_PAY', 'ACCTS_REC') );

```

59.127 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

Parameter	Description
p_user_name	The target user name.
p_granted_group_names	The names of groups to grant to p_user_name.

Example

This example creates a user group (MANAGER) and a user (Example User) and then grants MANAGER to Example User.

```
apex_util.create_user_group (
    p_group_name => 'MANAGER' );
apex_util.create_user (
    p_user_name => 'Example User',
    p_web_password => 1_random_password );
-- grant MANAGER to Example User
apex_util.set_group_user_grants (
    p_user_name => 'Example User',
    p_granted_group_names => apex_t_varchar2('MANAGER'));
```

59.128 SET_LAST_NAME Procedure

 **Note:**

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with APEX Accounts authentication.

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

Parameter	Description
p_userid	The numeric ID of the user account.
p_last_name	LAST_NAME value to be saved in the user account.

Example

The following example shows how to use the SET_LAST_NAME procedure to set the value of LAST_NAME to 'SMITH' for the user 'FRANK'.

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

See Also:

- [GET_LAST_NAME Function](#)
- [GET_USER_ID Function](#)

59.129 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

```
APEX_UTIL.SET_PARSING_SCHEMA_FOR_REQUEST (
    p_schema IN VARCHAR2 );
```

Parameters

Parameter	Description
p_schema	The new parsing schema.

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;
```

59.130 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

Parameter	Description
p_preference	Name of the preference (case-sensitive).
p_value	Value of the preference.
p_user	User for whom the preference is being set.

Example

The following example shows how to use the SET_PREFERENCE procedure to set a preference called 'default_view' to the value 'WEEKLY' that persists beyond session for the currently authenticated user.

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

See Also:

- [GET_PREFERENCE Function](#)
- [REMOVE_PREFERENCE Procedure](#)

59.131 SET_SECURITY_GROUP_ID Procedure

Use this procedure with `apex_util.find_security_group_id` to ease the use of the mail package in batch mode. This procedure is especially useful when a schema is associated with more than one workspace. For example, you might want to create a procedure that is run by a nightly job to email all outstanding tasks.

Syntax

```
APEX_UTIL.SET_SECURITY_GROUP_ID (
    p_security_group_id    IN NUMBER);
```

Parameters

Parameter	Description
p_security_group_id	This is the security group id of the workspace you are working in.

Example

The following example sends an alert to each user that has had a task assigned within the last day.

```
create or replace procedure new_tasks
is
    l_workspace_id      number;
    l_subject           varchar2(2000);
    l_body               clob;
    l_body_html          clob;
begin
    l_workspace_id := apex_util.find_security_group_id (p_workspace =>
'PROJECTS');
    apex_util.set_security_group_id (p_security_group_id => l_workspace_id);

    l_body := ' ';
    l_subject := 'You have new tasks';
    for c1 in (select distinct(p.email_address) email_address, p.user_id
                from teamsp_user_profile p, teamsp_tasks t
                where p.user_id = t.assigned_to_user_id
                  and t.created_on > sysdate - 1
                  and p.email_address is not null ) loop
        l_body_html := '<p />The following tasks have been added.';
        for c2 in (select task_name, due_date
                    from teamsp_tasks
                    where assigned_to_user_id = c1.user_id
                      and created_on > sysdate - 1 ) loop
            l_body_html := l_body_html || '<p />Task: '||c2.task_name||', due
'||c2.due_date;
        end loop;
    apex_mail.send (
        p_to      => c1.email_address,
        p_from    => c1.email_address,
        p_body    => l_body,
        p_body_html=> l_body_html,
        p_subj   => l_subject );
    end loop;
end;
```

59.132 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;
```

59.133 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;
```

59.134 SET_SESSION_LANG Procedure

This procedure sets the language for the current user in the current Oracle APEX session. The language must be a valid IANA language name.

Syntax

```
APEX_UTIL.SET_SESSION_LANG (
    p_lang IN VARCHAR2 );
```

Parameters

Parameter	Description
p_lang	This is an IANA language code. Examples include en, de, de-at, zh-cn, and pt-br.

Example

The following example sets the language for the current user for the duration of the APEX session.

```
BEGIN
    APEX_UTIL.SET_SESSION_LANG( P_LANG => 'en' );
END;
```

59.135 SET_SESSION_LIFETIME_SECONDS Procedure

This procedure sets the current session's Maximum Session Length in Seconds value, overriding the corresponding application attribute. This enables 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

Parameter	Description
p_seconds	A positive integer indicating the number of seconds that the session used by the application can exist.
p_scope	This parameter is obsolete. The procedure always sets the lifetime for the whole session.

Example 1

The following example sets the current application's Maximum Session Length in Seconds attribute to 7200 seconds (two hours).

By setting 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 APEX 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 sets 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;
```

59.136 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

Parameter	Description
p_seconds	A positive integer indicating the number of seconds allowed between page requests.
p_scope	This parameter is obsolete. The procedure always sets the lifetime for the whole session.

Example 1

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 1200 seconds (twenty minutes). The following example applies to all applications using the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 1200);
END;
```

Example 2

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 600 seconds (ten minutes). This example applies to all applications using the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 600);
END;
```

59.137 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;
```

59.138 SET_SESSION_SCREEN_READER_ON Procedure

This procedure puts the current session into screen reader mode.

Syntax

```
APEX_UTIL.SET_SESSION_SCREEN_READER_ON;
```

Parameters

None.

Example

In this example, the current session is put into screen reader mode.

```
BEGIN
    apex_util.set_session_screen_reader_on;
END;
```

59.139 SET_SESSION_STATE Procedure

This procedure sets session state for a current Oracle APEX 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

Parameter	Description
p_name	Name of the application-level or page-level item for which you are setting session state.
p_value	Value of session state to set.
p_commit	If TRUE (default), commit after modifying session state. If FALSE or if the existing value in session state equals p_value, no commit.
	This parameter is ignored when the application's Session State Changes attribute is set to End Of Request.

Example

The following example uses the SET_SESSION_STATE procedure to change 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_NUMERIC_SESSION_STATE Function](#)
- [GET_SESSION_STATE Function](#)
- Understanding Session State Management in *Oracle APEX App Builder User's Guide*

59.140 SET_SESSION_TERRITORY Procedure

This procedure sets the territory to be used for the current user in the current Oracle APEX session. The territory name must be a valid Oracle territory.

Syntax

```
APEX_UTIL.SET_SESSION_TERRITORY (
    p_territory IN VARCHAR2 );
```

Parameters

Parameter	Description
p_territory	A valid Oracle territory name. Examples include: AMERICA, UNITED KINGDOM, ISRAEL, AUSTRIA, and UNITED ARAB EMIRATES.

Example

The following example shows how to use the SET_SESSION_TERRITORY procedure. It sets the territory for the current user for the duration of the APEX session.

```
BEGIN
    APEX_UTIL.SET_SESSION_TERRITORY( P_TERRITORY => 'UNITED KINGDOM');
END;
```

59.141 SET_SESSION_TIME_ZONE Procedure

This procedure sets the time zone to be used for the current user in the current Oracle APEX session.

Syntax

```
APEX_UTIL.SET_SESSION_TIME_ZONE (
    p_time_zone IN VARCHAR2 );
```

Parameters

Parameter	Description
p_timezone	A time zone value in the form of hours and minutes. Examples include: +09:00, 04:00, -05:00.

Example

The following example shows how to use the SET_SESSION_TIME_ZONE procedure. It sets the time zone for the current user for the duration of the APEX session.

```
BEGIN
    APEX_UTIL.SET_SESSION_TIME_ZONE( P_TIME_ZONE => '-05:00');
END;
```

59.142 SET_USERNAME Procedure

Note:

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with APEX Accounts authentication.

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

Parameter	Description
p_userid	The numeric ID of the user account.
p_username	USER_NAME value to be saved in the user account.

Example

The following example shows how to use the SET_USERNAME procedure to set the value of USERNAME to 'USER-XRAY' for the user 'FRANK'.

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

See Also:

- [GET_USERNAME Function](#)
- [GET_USER_ID Function](#)

59.143 SET_WORKSPACE Procedure

This procedure sets the current workspace.

Syntax

```
APEX_UTIL.SET_WORKSPACE (
    p_workspace IN VARCHAR2 )
```

Parameters

Parameters	Description
p_workspace	The workspace's short name.

Example

This example sets the workspace MY_WORKSPACE.

```
apex_util.set_workspace (
    p_workspace => 'MY_WORKSPACE' );
```

59.144 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 procedure returns a link that when clicked switches the high contrast mode on.

Syntax

```
APEX_UTIL.SHOW_HIGH_CONTRAST_MODE_TOGGLE (
    p_on_message  IN VARCHAR2 DEFAULT NULL,
    p_off_message IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameters	Description
p_on_message	Optional text used for the link to switch to high contrast mode when you are in standard mode. If this parameter is not passed, the default "Set High Contrast Mode On" text displays.
p_off_message	Optional text used for the link to switch to standard mode when you are in high contrast mode. If this parameter is not passed, the default "Set High Contrast Mode Off" text displays.

Example

When running in standard mode, this procedure displays a link, Set High Contrast Mode On, that when clicked refreshes the current page and switches on high contrast mode. When running in high contrast mode, a link, Set High Contrast Mode Off, is displayed, that refreshes the current page and switches back to standard mode when clicked.

```
BEGIN
    apex_util.show_high_contrast_mode_toggle;
END;
```

Note:

There are also 2 translatable system messages that can be overridden at application level to change the default link text that is returned for this toggle. They include:

- APEX.SET_HIGH_CONTRAST_MODE_OFF - Default text = Set High Contrast Mode Off
- APEX.SET_HIGH_CONTRAST_MODE_ON - Default text = Set High Contrast Mode On

 See Also:

[GET_HIGH_CONTRAST_MODE_TOGGLE Function](#)

59.145 SHOW_SCREEN_READER_MODE_TOGGLE Procedure

This procedure displays a link to the current page to turn on or off (or toggle) the mode. For example, if you are in standard mode, this procedure 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

Parameter	Description
p_on_message	Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text is displayed.
p_off_message	Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text is displayed.

Example

When running in standard mode, this procedure displays a link 'Set Screen Reader Mode On', that when clicked refreshes the current page and switches on screen reader mode. When running in screen reader mode, a link with the text 'Set Screen Reader Mode Off' displays. Clicking the link refreshes the current page and switches back to standard mode.

```
BEGIN
    apex_util.show_screen_reader_mode_toggle;
END;
```

59.146 STRING_TO_TABLE Function (Deprecated)

 Note:

This function is deprecated. Oracle recommends APEX_STRING.STRING_TO_TABLE instead.

See [STRING_TO_TABLE Function](#).

Given a string, this function returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2. This array is a VARCHAR2 (32767) table.

Syntax

```
APEX_UTIL.STRING_TO_TABLE (
    p_string      IN VARCHAR2,
    p_separator   IN VARCHAR2 DEFAULT ':')
RETURN APEX_APPLICATION_GLOBAL.VC_ARR2;
```

Parameters

Parameter	Description
p_string	String to be converted into a PL/SQL table of type APEX_APPLICATION_GLOBAL.VC_ARR2.
p_separator	String separator. The default is a colon.

Example

The following example demonstrates how the function is passed the string One:Two:Three in the p_string parameter and returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2 containing three elements: the element at position 1 contains the value One, position 2 contains the value Two, and position 3 contains the value Three. This is then output using the HTP.P function call.

```
DECLARE
    l_vc_arr2      APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_vc_arr2 := APEX_UTIL.STRING_TO_TABLE('One:Two:Three');
    FOR z IN 1..l_vc_arr2.count LOOP
        htp.p(l_vc_arr2(z));
    END LOOP;
END;
```

See Also:

- [STRING_TO_TABLE Function](#)
- [TABLE_TO_STRING Function \(Deprecated\)](#)
- [SPLIT Function Signature 1](#)
- [SPLIT Function Signature 2](#)
- [SPLIT_NUMBERS Function](#)

59.147 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 APEX 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

Parameter	Description
p_username	Username that identifies the account in the current workspace.
p_password	Password to be checked against password strength rules.
p_old_password	Current password for the account. Used only to enforce "new password must differ from old" rule.
p_workspace_name	Current workspace name, used only to enforce "password must not contain workspace name" rule.
p_use_strong_rules	Pass FALSE when calling this API.
p_min_length_err	Result returns TRUE or FALSE depending upon whether the password meets minimum length requirement.
p_new_differs_by_err	Result returns TRUE or FALSE depending upon whether the password meets "new password must differ from old" requirements.
p_one_alpha_err	Result returns TRUE or FALSE depending upon whether the password meets requirement to contain at least one alphabetic character.
p_one_numeric_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one numeric character.
p_one_punctuation_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one punctuation character.
p_one_upper_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one upper-case character.
p_one_lower_err	Result returns TRUE or FALSE depending upon whether the password meets requirements to contain at least one lower-case character.
p_not_like_username_err	Result returns TRUE or FALSE depending upon whether the password meets requirements that it must not contain the username.
p_not_like_workspace_name_err	Result returns TRUE or FALSE depending upon whether the password meets requirements that it must not contain the workspace name.
p_not_like_words_err	Result returns TRUE or FALSE whether the password meets requirements that it must not contain specified simple words.

Parameter	Description
p_not_reusable_err	Result returns TRUE or FALSE whether the password can be reused based on password history rules.

Example

The following example checks if the new password `foo` for the user `SOMEBODY` meets all the password strength requirements defined by the APEX 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 APEX site administrator defined that passwords must have at least one numeric character and the password `foo` is checked, then the `p_one_numeric_err` `OUT` parameter returns `TRUE` and the message "Password must contain at least one numeric character" displays.

```

DECLARE
    l_username          varchar2(30);
    l_password          varchar2(30);
    l_old_password      varchar2(30);
    l_workspace_name    varchar2(30);
    l_min_length_err    boolean;
    l_new_differs_by_err boolean;
    l_one_alpha_err     boolean;
    l_one_numeric_err   boolean;
    l_one_punctuation_err boolean;
    l_one_upper_err     boolean;
    l_one_lower_err     boolean;
    l_not_like_username_err boolean;
    l_not_like_workspace_name_err boolean;
    l_not_like_words_err boolean;
    l_not_reusable_err  boolean;
    l_password_history_days pls_integer;
BEGIN
    l_username := 'SOMEBODY';
    l_password := 'foo';
    l_old_password := 'foo';
    l_workspace_name := 'XYX_WS';
    l_password_history_days :=
        apex_instance_admin.get_parameter ('PASSWORD_HISTORY_DAYS');

    APEX_UTIL.STRONG_PASSWORD_CHECK(
        p_username          => l_username,
        p_password          => l_password,
        p_old_password      => l_old_password,
        p_workspace_name    => l_workspace_name,
        p_use_strong_rules => false,
        p_min_length_err    => l_min_length_err,
        p_new_differs_by_err=> l_new_differs_by_err,
        p_one_alpha_err     => l_one_alpha_err,
        p_one_numeric_err   => l_one_numeric_err,
        p_one_punctuation_err=> l_one_punctuation_err,
        p_one_upper_err     => l_one_upper_err,
        p_one_lower_err     => l_one_lower_err,
        p_not_like_username_err=> l_not_like_username_err,
        p_not_like_workspace_name_err=> l_not_like_workspace_name_err,

```

```
p_not_like_words_err          => l_not_like_words_err,  
p_not_reusable_err           => l_not_reusable_err);  
  
IF l_min_length_err THEN  
    http.p('Password is too short');  
END IF;  
  
IF l_new_differs_by_err THEN  
    http.p('Password is too similar to the old password');  
END IF;  
  
IF l_one_alpha_err THEN  
    http.p('Password must contain at least one alphabetic character');  
END IF;  
  
IF l_one_numeric_err THEN  
    http.p('Password must contain at least one numeric character');  
END IF;  
  
IF l_one_punctuation_err THEN  
    http.p('Password must contain at least one punctuation character');  
END IF;  
  
IF l_one_upper_err THEN  
    http.p('Password must contain at least one upper-case character');  
END IF;  
  
IF l_one_lower_err THEN  
    http.p('Password must contain at least one lower-case character');  
END IF;  
  
IF l_not_like_username_err THEN  
    http.p('Password may not contain the username');  
END IF;  
  
IF l_not_like_workspace_name_err THEN  
    http.p('Password may not contain the workspace name');  
END IF;  
  
IF l_not_like_words_err THEN  
    http.p('Password contains one or more prohibited common words');  
END IF;  
  
IF l_not_reusable_err THEN  
    http.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 APEX Administration Guide](#)

59.148 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 APEX site administrator.

Syntax

```
APEX_UTIL.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

Parameter	Description
p_username	Username that identifies the account in the current workspace.
p_password	Password to be checked against password strength rules.
p_old_password	Current password for the account. Used only to enforce "new password must differ from old" rule.
p_workspace_name	Current workspace name, used only to enforce "password must not contain workspace name" rule.

Example

The following example checks the new password `foo` for the user `SOMEBOY` meets all the password strength requirements defined by the APEX site administrator. If any of the checks fail, then the example outputs formatted HTML showing details of where the new password fails to meet requirements.

```
DECLARE
    l_username          varchar2(30);
    l_password           varchar2(30);
    l_old_password       varchar2(30);
    l_workspace_name     varchar2(30);
BEGIN
    l_username := 'SOMEBOY';
    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;
```

59.149 SUBMIT_FEEDBACK Procedure

This procedure enables you to write a procedure to submit feedback, rather than using the feedback page generated by Create Page Wizard.

Syntax

```
APEX_UTIL.SUBMIT_FEEDBACK (
    p_comment          IN VARCHAR2 DEFAULT NULL,
    p_type              IN NUMBER    DEFAULT '1',
    p_application_id   IN VARCHAR2 DEFAULT NULL,
    p_page_id           IN VARCHAR2 DEFAULT NULL,
    p_email              IN VARCHAR2 DEFAULT NULL,
    p_screen_width      IN VARCHAR2 DEFAULT NULL,
    p_screen_height     IN VARCHAR2 DEFAULT NULL,
    p_attribute_01       IN VARCHAR2 DEFAULT NULL,
    p_attribute_02       IN VARCHAR2 DEFAULT NULL,
    p_attribute_03       IN VARCHAR2 DEFAULT NULL,
    p_attribute_04       IN VARCHAR2 DEFAULT NULL,
    p_attribute_05       IN VARCHAR2 DEFAULT NULL,
    p_attribute_06       IN VARCHAR2 DEFAULT NULL,
    p_attribute_07       IN VARCHAR2 DEFAULT NULL,
    p_attribute_08       IN VARCHAR2 DEFAULT NULL,
    p_label_01            IN VARCHAR2 DEFAULT NULL,
    p_label_02            IN VARCHAR2 DEFAULT NULL,
    p_label_03            IN VARCHAR2 DEFAULT NULL,
    p_label_04            IN VARCHAR2 DEFAULT NULL,
    p_label_05            IN VARCHAR2 DEFAULT NULL,
    p_label_06            IN VARCHAR2 DEFAULT NULL,
    p_label_07            IN VARCHAR2 DEFAULT NULL,
    p_label_08            IN VARCHAR2 DEFAULT NULL,
    p_rating             IN NUMBER    DEFAULT NULL,
    p_attachment_name    IN VARCHAR2 DEFAULT NULL ) ;
```

Parameters

Parameter	Description
p_comment	Comment to be submitted.
p_type	Type of feedback (1 is General Comment, 2 is Enhancement Request, 3 is Bug).
p_application_id	ID of application related to the feedback.
p_page_id	ID of page related to the feedback.
p_email	Email of the user providing the feedback.
p_screen_width	Width of screen at time feedback was provided.
p_screen_height	Height of screen at time feedback was provided.
p_attribute_01	Custom attribute for collecting feedback.
p_attribute_02	Custom attribute for collecting feedback.
p_attribute_03	Custom attribute for collecting feedback.
p_attribute_04	Custom attribute for collecting feedback.
p_attribute_05	Custom attribute for collecting feedback.

Parameter	Description
p_attribute_06	Custom attribute for collecting feedback.
p_attribute_07	Custom attribute for collecting feedback.
p_attribute_08	Custom attribute for collecting feedback.
p_label_01	Label for corresponding custom attribute.
p_label_02	Label for corresponding custom attribute.
p_label_03	Label for corresponding custom attribute.
p_label_04	Label for corresponding custom attribute.
p_label_05	Label for corresponding custom attribute.
p_label_06	Label for corresponding custom attribute.
p_label_07	Label for corresponding custom attribute.
p_label_08	Label for corresponding custom attribute.
p_rating	User experience (3 is Good, 2 is Neutral, 1 is Bad).
p_attachment_name	Bind variable reference to the feedback form's "File Browse" page item.

Example

The following example submits a bad user experience because of a bug on page 22 within application 283.

```
BEGIN
    apex_util.submit_feedback (
        p_comment => 'This page does not render properly for me',
        p_type => 3,
        p_rating => 1,
        p_application_id => 283,
        p_page_id => 22,
        p_email => 'user@xyz.corp',
        p_attribute_01 => 'Charting',
        p_label_01 => 'Component' );
END;
/
```

59.150 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

Parameter	Description
p_feedback_id	ID of feedback that this is a follow up to.

Parameter	Description
p_follow_up	Text of follow up.
p_email	Email of user providing the follow up.

Example

The following example submits follow up to a previously filed feedback.

```
begin
    apex_util.submit_feedback_followup (
        p_feedback_id      => 12345,
        p_follow_up        => 'I tried this on another instance and it does not
work there either',
        p_email            => 'user@xyz.corp' );
end;
/
```

59.151 TABLE_TO_STRING Function (Deprecated)

**Note:**

This function is deprecated. Oracle recommends using the `JOIN` and `JOIN_CLOB` functions instead.

Given a PL/SQL table of type `APEX_APPLICATION_GLOBAL.VC_ARR2`, this function returns a delimited string separated by the supplied separator, or by the default separator, a colon (:).

Syntax

```
APEX_UTIL.TABLE_TO_STRING (
    p_table      IN      apex_application_global.vc_arr2,
    p_string     IN      VARCHAR2 DEFAULT ':')
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_string	String separator. Default separator is a colon (:).
p_table	PL/SQL table that is to be converted into a delimited string.

Example

The following example returns a comma delimited string of contact names that are associated with the provided `cust_id`.

```
create or replace function get_contacts (
    p_cust_id  in  number )
return varchar2
```

```
is
    l_vc_arr2    apex_application_global.vc_arr2;
    l_contacts   varchar2(32000);

BEGIN

    select contact_name
        bulk collect
        into l_vc_arr2
        from contacts
    where cust_id = p_cust_id
        order by contact_name;

    l_contacts := apex_util.table_to_string (
        p_table => l_vc_arr2,
        p_string => ', ');

    return l_contacts;

END get_contacts;
```

 **See Also:**

- [STRING_TO_TABLE Function \(Deprecated\)](#)
- [JOIN Function Signature 1](#)
- [JOIN Function Signature 2](#)
- [JOIN_CLOB Function](#)

59.152 UNEXPIRE_END_USER_ACCOUNT Procedure

 **Note:**

This procedure operates on the native Oracle APEX user accounts repository and is only applicable to applications configured with APEX Accounts authentication.

This procedure makes expired end users accounts and the associated passwords usable, enabling an end user to log into developed applications.

Syntax

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

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example renews (unexpires) an APEX 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 into 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:

- [EXPIRE_END_USER_ACCOUNT Procedure](#)
- [END_USER_ACCOUNT_DAYS_LEFT Function](#)

59.153 UNEXPIRE_WORKSPACE_ACCOUNT Procedure

This procedure unexpires developer and workspace administrator accounts and the associated passwords, enabling the developer or administrator to log into a workspace.

Syntax

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

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example shows how to use the UNEXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to renew (unexpire) an APEX workspace administrator account in the current workspace. This action specifically renews the account for use by developers or administrators

to log into a workspace and may also renew the account for its use by end users to authenticate to developed applications.

This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (select user_name from apex_users) loop
        APEX_UTIL.UNEXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```

See Also:

- [EXPIRE_WORKSPACE_ACCOUNT Procedure](#)
- [WORKSPACE_ACCOUNT_DAYS_LEFT Function](#)

59.154 UNLOCK_ACCOUNT Procedure

This procedure sets a user account status to unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.UNLOCK_ACCOUNT (
    p_user_name IN VARCHAR2 )
```

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example unlocks all Oracle APEX accounts (workspace administrator, developer, or end user) in the current workspace.

```
BEGIN
    FOR c1 IN (SELECT user_name from apex_workspace_apex_users) LOOP
        APEX_UTIL.UNLOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('User Account:'||c1.user_name||' is now unlocked.');
    END LOOP;
END;
```

 **See Also:**

- [LOCK_ACCOUNT Procedure](#)
- [GET_ACCOUNT_LOCKED_STATUS Function](#)

59.155 URL_ENCODE Function (Deprecated)

 **Note:**

This API is deprecated and will be removed in a future release.

Use the UTL_URL.ESCAPE function instead.

The following special characters are encoded as follows:

Special Characters	After Encoding
%	%25
+	%2B
space	+
.	%2E
*	%2A
?	%3F
\	%5C
/	%2F
>	%3E
<	%3C
}	%7B
{	%7D
~	%7E
[%5B
]	%5D
;	%3B
?	%3F
@	%40
&	%26
#	%23
	%7C
^	%5E
:	%3A
=	%3D
\$	%24

Syntax

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

Parameters

Parameter	Description
p_url	The string to be encoded.

Example

The following example shows how to use the `URL_ENCODE` function.

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

In this example, the following URL:

`http://www.example.com?id=1&cat=foo`

Would be returned as:

`http%3A%2F%2Fwww%2Eexample%2Ecom%3Fid%3D1%26cat%3Dfoo`

 **See Also:**

`UTL_URL.ESCAPE` in *Oracle Database PL/SQL Packages and Types Reference*

59.156 WORKSPACE_ACCOUNT_DAYS_LEFT Function

This function returns the number of days remaining before the developer or workspace administrator account password expires. Any authenticated user can run this function in a page request context.

Syntax

```
APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT (
    p_user_name IN VARCHAR2 )
RETURN NUMBER;
```

Parameters

Parameter	Description
p_user_name	The user name of the user account.

Example

The following example finds the number of days remaining before an Oracle APEX administrator or developer account in the current workspace expires.

```
DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from apex_workspace_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 APEX by enabling 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. Functions parse the responses from Web Services and 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](#)
- [About Web Credentials and APEX_WEB_SERVICE](#)
- [Data Types](#)
- [Global Variables](#)
- [APPEND_TO_MULTIPART Procedure Signature 1](#)
- [APPEND_TO_MULTIPART Procedure Signature 2](#)
- [BLOB2CLOBBASE64 Function](#)
- [CLEAR_REQUEST_COOKIES Procedure](#)
- [CLEAR_REQUEST_HEADERS Procedure](#)
- [CLOBBASE642BLOB Function](#)
- [GENERATE_REQUEST_BODY Function](#)
- [GET_REQUEST_HEADER Function](#)
- [MAKE_REQUEST Function Signature 1](#)
- [MAKE_REQUEST Function Signature 2](#)
- [MAKE_REQUEST Procedure Signature 1](#)
- [MAKE_REQUEST Procedure Signature 2](#)
- [MAKE_REST_REQUEST Function](#)
- [MAKE_REST_REQUEST_B Function](#)
- [OAUTH_AUTHENTICATE_CREDENTIAL Procedure](#)
- [OAUTH_AUTHENTICATE Procedure Signature 1](#)
- [OAUTH_AUTHENTICATE Procedure Signature 2 \(Deprecated\)](#)
- [OAUTH_GET_LAST_TOKEN Function](#)
- [OAUTH_SET_TOKEN Procedure](#)
- [PARSE_RESPONSE Function](#)

- [PARSE_RESPONSE_CLOB Function](#)
- [PARSE_XML Function](#)
- [PARSE_XML_CLOB Function](#)
- [REMOVE_REQUEST_HEADER Procedure](#)
- [SET_REQUEST_ECID_CONTEXT Procedure](#)
- [SET_REQUEST_HEADERS Procedure](#)

60.1 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 Oracle APEX.

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 want to invoke a Web service as part of a validation.
- When you need to pass a large binary parameter to a Web service that is base64 encoded.
- [Invoking a SOAP-style Web Service](#)
- [Invoking a RESTful-style Web Service](#)
- [Setting Cookies and HTTP Headers](#)
- [Retrieving Cookies and HTTP Headers](#)

60.1.1 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 `XMLETYPE`.

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 `XMLETYPE`.

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`.

Example

The following is an example of using the `BLOB2CLOBBASE64` function to encode a parameter, the `MAKE_REQUEST` procedure to call a Web service, and the `PARSE_RESPONSE` function to extract a specific value from the response.

```
DECLARE
  l_filename varchar2(255);
  l_BLOB BLOB;
```

```

l_CLOB CLOB;
l_envelope CLOB;
l_response_msg varchar2(32767);
BEGIN
    IF :P1_FILE IS NOT NULL THEN
        SELECT filename, BLOB_CONTENT
        INTO l_filename, l_BLOB
        FROM APEX_APPLICATION_FILES
        WHERE name = :P1_FILE;

        l_CLOB := apex_web_service.blob2clobbase64(l_BLOB);

        l_envelope := q'!<?xml version='1.0' encoding='UTF-8'?>!';
        l_envelope := l_envelope|| '<soapenv:Envelope xmlns:soapenv="http://
schemas.xmlsoap.org/soap/envelope/" xmlns:chec="http://www.stellent.com/
CheckIn/">
            <soapenv:Header/>
            <soapenv:Body>
                <chec:CheckInUniversal>
                    <chec:dDocName'||l_filename||'|</chec:dDocName>
                    <chec:dDocTitle'||l_filename||'|</chec:dDocTitle>
                    <chec:dDocType>Document</chec:dDocType>
                    <chec:dDocAuthor>GM</chec:dDocAuthor>
                    <chec:dSecurityGroup>Public</chec:dSecurityGroup>
                    <chec:dDocAccount></chec:dDocAccount>
                    <chec:CustomDocMetaData>
                        <chec:property>
                            <chec:name></chec:name>
                            <chec:value></chec:value>
                        </chec:property>
                    </chec:CustomDocMetaData>
                    <chec:primaryFile>
                        <chec:fileName'||l_filename'||</chec:fileName>
                        <chec:fileContent'||l_CLOB||'|</chec:fileContent>
                    </chec:primaryFile>
                    <chec:alternateFile>
                        <chec:fileName></chec:fileName>
                        <chec:fileContent></chec:fileContent>
                    </chec:alternateFile>
                    <chec:extraProps>
                        <chec:property>
                            <chec:name></chec:name>
                            <chec:value></chec:value>
                        </chec:property>
                    </chec:extraProps>
                </chec:CheckInUniversal>
            </soapenv:Body>
        </soapenv:Envelope>';

apex_web_service.make_request(
    p_url          => 'http://192.0.2.1/idc/idcplg',
    p_action        => 'http://192.0.2.1/CheckIn/',
    p_collection_name => 'STELLENT_CHECKIN',
    p_envelope      => l_envelope,
    p_username       => 'sysadmin',
    p_password       => 'password' );

```

```

l_response_msg := apex_web_service.parse_response(
    p_collection_name=>'STELLENT_CHECKIN',
p_xpath=> '//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/
idc:StatusInfo/idc:statusMessage/text()',
    p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn"' );

:P1_RES_MSG := l_response_msg;

END IF;
END;

```

60.1.2 Invoking a RESTful-style Web Service

RESTful-style Web services use a simpler architecture than SOAP. Often 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.

Example

The following is an example of MAKE_REST_REQUEST in an application process that is callable by Ajax.

```

DECLARE
    l_clob clob;
    l_buffer        varchar2(32767);
    l_amount        number;
    l_offset        number;
BEGIN

    l_clob := apex_web_service.make_rest_request(
        p_url => 'http://us.music.yahooapis.com/video/v1/list/
published/popular',
        p_http_method => 'GET',
        p_parm_name => apex_util.string_to_table('appid:format'),
        p_parm_value =>
apex_util.string_to_table(apex_application.g_x01||':'||
apex_application.g_x02));

    l_amount := 32000;
    l_offset := 1;
    BEGIN
        LOOP
            dbms_lob.read( l_clob, l_amount, l_offset, l_buffer );
            http.p(l_buffer);
            l_offset := l_offset + l_amount;
            l_amount := 32000;
        END LOOP;
    EXCEPTION
        WHEN no_data_found THEN
            NULL;
    END;

END;

```

60.1.3 Setting Cookies and HTTP Headers

Set cookies and HTTP headers that should be sent along with a Web Service request by populating the globals `g_request_cookies` and `g_request_headers` before the process that invokes the Web Service.

The following example populates 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;
```

See Also:

- [Global Variables](#)
- [Retrieving Cookies and HTTP Headers](#)

60.1.4 Retrieving Cookies and HTTP Headers

When you invoke a Web service using any of the supported methods in Oracle APEX, 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.

When you invoke a Web service using any of the supported methods in APEX, the `g_status_code` global is populated with the numeric HTTP status code of the response (for example, 200 or 404). The `g_response_cookies` and `g_headers` globals are populated if the Web service response included any cookies or HTTP headers.

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;

```

See Also:

- [Global Variables](#)
- [Setting Cookies and HTTP Headers](#)

60.2 About Web Credentials and APEX_WEB_SERVICE

You can use the `MAKE_REQUEST` and `MAKE_REST_REQUEST` procedures to enable Web Credentials in order to authenticate against the remote Web Service.

Web Credentials can be used with the `APEX_WEB_SERVICE` package from outside the context of an Oracle APEX application (such as from SQLcl or from a Database Scheduler job) as long as the database user making the call is mapped to an APEX workspace.

If the database user is mapped to multiple workspaces, you must first call `APEX_UTIL.SET_WORKSPACE` or `APEX_UTIL.SET_SECURITY_GROUP_ID` as in the following examples.

If the database user is mapped to multiple workspaces, you must first call `APEX_UTIL.SET_WORKSPACE` or `APEX_UTIL.SET_SECURITY_GROUP_ID` as in the following examples. The `APEX_WEB_SERVICE` package cannot be used by database users that are not mapped to any workspace unless they have been granted the role `APEX_ADMINISTRATOR_ROLE`.

Examples

Example 1

```
apex_util.set_workspace(p_workspace => 'MY_WORKSPACE');
```

Example 2

```
FOR c1 in (
    select workspace_id
        from apex_applications
       where application_id = 100 )
LOOP
    apex_util.set_security_group_id(p_security_group_id => c1.workspace_id);
END LOOP;
```

See Also:

Managing Web Credentials in *Oracle APEX App Builder User's Guide*.

60.3 Data Types

The `APEX_WEB_SERVICE` package uses the following data types.

```
type header is record (
    name varchar2(256),
    value varchar2(32767) );

type header_table is table of header index by binary_integer;
```

60.4 Global Variables

Global Variable	Data Type	Description
<code>g_headers</code>	<code>header_table</code>	Global holding HTTP headers to send with a HTTP request.

Global Variable	Data Type	Description
g_reason_phrase	varchar2(32767)	Reason phrase corresponding with the status code received with the HTTP response.
g_request_cookies	sys.utl_http.cookie_table	Global holding cookies to send with a HTTP request.
g_request_headers	header_table	Global holding HTTP headers received with the HTTP response.
g_response_cookies	sys.utl_http.cookie_table	Global holding cookies received with the HTTP response.
g_status_code	pls_integer	Status code received from the HTTP request.

The `g_status_code` and `g_reason_phrase` globals are set after each HTTP request so that you can get its outcome (for example, 200=OK, 400=Bad Request).

See Also:

- [Setting Cookies and HTTP Headers](#)
- [Retrieving Cookies and HTTP Headers](#)

60.5 APPEND_TO_MULTIPART Procedure Signature 1

This procedure adds a BLOB to a multipart/form request body.

Syntax

```
APEX_WEB_SERVICE.APPEND_TO_MULTIPART (
    p_multipart      IN OUT NOCOPY t_multipart_parts,
    p_name           IN          VARCHAR2,
    p_filename       IN          VARCHAR2 DEFAULT NULL,
    p_content_type   IN          VARCHAR2 DEFAULT 'application/octet-stream',
    p_body_blob      IN          BLOB );
```

Parameters

Parameter	Description
p_multipart	The table type for the multipart/request body, <code>t_multipart_parts</code> .
p_name	The name of the multipart data.
p_filename	The filename of the multipart data if it exists.
p_content_type	The content type of the multipart data.
p_body_blob	The content to add in BLOB.

Example

```

DECLARE
    l_multipart      apex_web_service.t_multipart_parts;
BEGIN
    apex_web_service.append_to_multipart (
        p_multipart      => l_multipart,
        p_name           => 'param1',
        p_content_type   => 'application/octet-stream',
        p_body_blob      => (select blob from table where id = 1) );
END;

```

60.6 APPEND_TO_MULTIPART Procedure Signature 2

This procedure adds a CLOB to a multipart/form request body.

Syntax

```
APEX_WEB_SERVICE.APPEND_TO_MULTIPART (
    p_multipart      IN OUT NOCOPY t_multipart_parts,
    p_name           IN          VARCHAR2,
    p_filename       IN          VARCHAR2 DEFAULT NULL,
    p_content_type   IN          VARCHAR2 DEFAULT 'application/octet-stream',
    p_body           IN          CLOB );
```

Parameters

Parameter	Description
p_multipart	The table type for the multipart/request body, t_multipart_parts.
p_name	The name of the multipart data.
p_filename	The filename of the multipart data if it exists.
p_content_type	The content type of the multipart data.
p_body	The content to add in CLOB.

Example

```

DECLARE
    l_multipart apex_web_service.t_multipart_parts;
BEGIN
    apex_web_service.append_to_multipart (
        p_multipart      => l_multipart,
        p_name           => 'param1',
        p_content_type   => 'application/json',
        p_body           => to_clob( '{"hello":"world"}' ) );
END;

```

60.7 BLOB2CLOBBASE64 Function

This function converts a BLOB data type 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,
    p_newlines  IN VARCHAR2 DEFAULT 'Y',
    p_padding   IN VARCHAR2 DEFAULT 'N' )
RETURN CLOB;
```

Parameters

Parameter	Description
p_blob	The BLOB to convert into base64-encoded CLOB.
p_newlines	Whether the generated base64 content contains line breaks.
p_padding	Whether to pad the generated base64 content with "=" so that the length becomes a multiple of 4.

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;
```

60.8 CLEAR_REQUEST_COOKIES Procedure

This procedure clears all cookies, so that the next MAKE_REST_REQUEST call executes without sending any cookies. This procedure clears the cookie globals in APEX_WEB_SERVICE and in UTL_HTTP.

Syntax

```
APEX_WEB_SERVICE.CLEAR_REQUEST_COOKIES;
```

Parameters

None.

Example

```
begin
    apex_web_service.clear_request_cookies;
end;
```

60.9 CLEAR_REQUEST_HEADERS Procedure

This procedure clears the current request headers.

Syntax

```
APEX_WEB_SERVICE.CLEAR_REQUEST_HEADERS;
```

Parameters

None.

Example

```
begin
    apex_web_service.clear_request_headers;
end;
```

60.10 CLOBBASE642BLOB Function

This function converts 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

Parameter	Description
<code>p_clob</code>	The <code>base64</code> -encoded <code>CLOB</code> to convert into a <code>BLOB</code> .

Example

The following example retrieves a `base64`-encoded node from an XML document as a `CLOB` and converts it into a `BLOB`.

```
DECLARE
    l_base64 CLOB;
    l_blob    BLOB;
    l_xml     XMLTYPE;
BEGIN
```

```

l_base64 := apex_web_service.parse_xml_clob(l_xml, ' //runReportReturn/
reportBytes/text()');
l_blob := apex_web_service.clobbase642blob(l_base64);
END;

```

60.11 GENERATE_REQUEST_BODY Function

This function generates the multipart/form-data request body from the data in the `t_multipart` array.

Syntax

```
APEX_WEB_SERVICE.GENERATE_REQUEST_BODY(
    p_multipart      IN t_multipart_parts,
    p_to_charset     IN VARCHAR2 DEFAULT www_flow_lang.get_db_charset )
RETURN BLOB;
```

Parameters

Parameter	Description
<code>p_multipart</code>	The table type for the multipart/request body, <code>t_multipart_parts</code> .
<code>p_to_charset</code>	The target character set for the parts that are CLOBs. This parameter defaults to the current character set of the database.

Examples

This example stores the multipart/form request in a local BLOB variable.

```

DECLARE
    l_multipart      apex_web_service.t_multipart_parts;
    l_request_blob blob;
BEGIN
    l_request_blob := apex_web_service.generate_request_body (
                      p_multipart      => l_multipart );
END;

```

60.12 GET_REQUEST_HEADER Function

This function gets a specific request header value out of the request headers array.

Syntax

```
APEX_WEB_SERVICE.GET_REQUEST_HEADER (
    p_header_name    VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_header_name	The parameter name (case is normalized for the search in the header array).

Example

```
select apex_web_service.get_request_header('ECID-Context') from dual;
```

60.13 MAKE_REQUEST Function Signature 1

This function invokes 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 )
RETURN sys.xmltype;
```

Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation to be invoked.
p_version	The SOAP version (1.1 or 1.2). The default is 1.1.
p_envelope	The SOAP envelope to post to the service.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service.
p_scheme	The authentication scheme. Basic (default), AWS, Digest, or OAUTH_CLIENT_CRED if supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.

Parameter	Description
p_wallet_path	The file system path to a wallet if the URL endpoint is HTTPS. For example, file:/usr/home/oracle/WALLETS
p_wallet_pwd	The wallet path provided overrides the wallet defined in the instance settings.
p_https_host	The password to access the wallet.

Returns

The SOAP service response in an XMLTYPE.

Example 1

The following example invokes a SOAP-style Web service that returns movie listings. The result is stored in an XMLTYPE.

```

DECLARE
    l_envelope CLOB;
    l_xml       XMLTYPE;
BEGIN
    l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
        <tns:GetTheatersAndMovies>
            <tns:zipCode>43221</tns:zipCode>
            <tns:radius>5</tns:radius>
        </tns:GetTheatersAndMovies>
    </soap:Body>
</soap:Envelope>';

    l_xml := apex_web_service.make_request(
        p_url => 'http://www.ignyte.com/webservices/
ignyte.whatsshowing.webservice/moviefunctions.asmx',
        p_action => 'http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
        p_envelope => l_envelope
    );
END;

```

Example 2

This example invokes a SOAP service returning an XMLTYPE.

```

DECLARE
    l_xml sys.xmltype;
BEGIN
    l_xml := apex_web_service.make_request(
        p_url      => 'http://{host}:{port}/path/to/soap/service/',
        p_action   => 'doSoapRequest',

```

```

    p_envelope      => '{SOAP envelope in XML format}' );
END;

```

60.14 MAKE_REQUEST Function Signature 2

This function invokes a Web service with the supplied SOAP envelope and returns the response as 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_credential_static_id IN VARCHAR2,
    p_token_url     IN VARCHAR2 DEFAULT NULL,
    --
    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 )
RETURN sys.xmltype;

```

Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation invoked.
p_version	The SOAP version (1.1 or 1.2). The default is 1.1.
p_envelope	The SOAP envelope to post to the service.
p_credential_static_id	The name of the Web Credentials to be used. Web Credentials are configured in Workspace Utilities.
p_token_url	The URL to retrieve the token from for token-based authentication flows (such as OAuth2).
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The filesystem path to a wallet if request is HTTPS. For example, file:/usr/home/oracle/WALLETS
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.

Returns

The SOAP service response as an XMLTYPE.

Example

The following example invokes a SOAP service returning an XMLTYPE.

```
DECLARE
    l_xml sys.xmltype;
BEGIN
    l_xml := apex_web_service.make_request(
        p_url          => 'http://{host}:{port}/path/to/soap/
service/',
        p_action       => 'doSoapRequest',
        p_envelope     => '{SOAP envelope in XML format}',
        p_credential_static_id => 'My_Credentials',
        p_token_url   => 'http://{host}:{port}/ords/scott/oauth/
token' );
END;
```

60.15 MAKE_REQUEST Procedure Signature 1

This procedure invokes a SOAP-style Web service with the supplied SOAP envelope and stores the results in a collection.

Syntax

```
APEX_WEB_SERVICE.MAKE_REQUEST (
    p_url          IN VARCHAR2,
    p_action       IN VARCHAR2 DEFAULT NULL,
    p_version      IN VARCHAR2 DEFAULT '1.1',
    p_collection_name IN VARCHAR2 DEFAULT NULL,
    p_envelope     IN CLOB,
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_password     IN VARCHAR2 DEFAULT NULL,
    p_scheme       IN VARCHAR2 DEFAULT 'Basic',
    p_proxy_override IN VARCHAR2 DEFAULT NULL,
    p_transfer_timeout IN NUMBER DEFAULT 180,
    p_wallet_path  IN VARCHAR2 DEFAULT NULL,
    p_wallet_pwd   IN VARCHAR2 DEFAULT NULL,
    p_https_host   IN VARCHAR2 DEFAULT NULL )
```

Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation to be invoked.
p_version	The SOAP version (1.1 or 1.2). The default is 1.1.
p_collection_name	The name of the collection to store the response.
p_envelope	The SOAP envelope to post to the service.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme. Basic (default), AWS, or Digest if supported by your database release.

Parameter	Description
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The file system path to a wallet if the URL endpoint is HTTPS. For example, file:/usr/home/oracle/WALLETS
	The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.

Example 1

The following example retrieves a list of movies from a SOAP-style Web service. The response is stored in an Oracle APEX collection named MOVIE_LISTINGS.

```

DECLARE
    l_envelope CLOB;
BEGIN
    l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
        <tns:GetTheatersAndMovies>
            <tns:zipCode>43221</tns:zipCode>
            <tns:radius>5</tns:radius>
        </tns:GetTheatersAndMovies>
    </soap:Body>
</soap:Envelope>';

apex_web_service.make_request(
    p_url          => 'http://www.ignyte.com/webservices/
ignyte.whatsshowing.webservice/moviefunctions.asmx',
    p_action       => 'http://www.ignyte.com/whatsshowing/
GetTheatersAndMovies',
    p_collection_name  => 'MOVIE_LISTINGS',
    p_envelope      => l_envelope
);

END;

```

Example 2

This example invokes a SOAP service and stores the response in a collection.

```

BEGIN
    apex_web_service.make_request(
        p_url          => 'http://{host}:{port}/path/to/soap/service/',
        p_collection_name => 'MY_RESPONSE_COLLECTION',
        p_action       => 'doSoapRequest',

```

```

    p_envelope      => '{SOAP envelope in XML format}' );
END;

```

60.16 MAKE_REQUEST Procedure Signature 2

This procedure invokes a Web service with the supplied SOAP envelope and stores the response in a collection.

Syntax

```

APEX_WEB_SERVICE.MAKE_REQUEST (
    p_url           IN VARCHAR2,
    p_action        IN VARCHAR2 DEFAULT NULL,
    p_version       IN VARCHAR2 DEFAULT '1.1',
    p_collection_name IN VARCHAR2 DEFAULT NULL,
    p_envelope      IN CLOB,
    --
    p_credential_static_id IN VARCHAR2,
    p_token_url     IN VARCHAR2 DEFAULT NULL,
    --
    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

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation invoked.
p_version	The SOAP version (1.1 or 1.2). The default is 1.1.
p_collection_name	The name of the collection to store the response.
p_envelope	The SOAP envelope to post to the service.
p_credential_static_id	The name of the Web Credentials to be used. Web Credentials are configured in Workspace Utilities.
p_token_url	For token-based authentication flows, the URL where to get the token from.
p_proxy_override	The proxy to use for the request.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The filesystem path to a wallet if request is HTTPS. For example, file:/usr/home/oracle/WALLETS
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.

Example

The following example invokes a SOAP service and stores the response in a collection.

```
BEGIN
    apex_web_service.make_request(
        p_url                  => 'http://{host}:{port}/path/to/soap/
service/',
        p_collection_name      => 'MY_RESPONSE_COLLECTION',
        p_action                => 'doSoapRequest',
        p_envelope              => '{SOAP envelope in XML format}',
        p_credential_static_id => 'My_Credentials',
        p_token_url             => 'http://{host}:{port}/ords/scott/oauth/
token' );
END;
```

60.17 MAKE_REST_REQUEST Function

Use this function to invoke a RESTful style Web service supplying either name value pairs, a character based payload or a binary payload and returning the response in a CLOB.

Syntax

```
APEX_WEB_SERVICE.MAKE_REST_REQUEST (
    p_url                      IN VARCHAR2,
    p_http_method               IN VARCHAR2,
    p_username                  IN VARCHAR2 DEFAULT NULL,
    p_password                  IN VARCHAR2 DEFAULT NULL,
    p_scheme                     IN VARCHAR2 DEFAULT 'Basic',
    p_proxy_override             IN VARCHAR2 DEFAULT NULL,
    p_transfer_timeout          IN NUMBER   DEFAULT 180,
    p_body                      IN CLOB     DEFAULT EMPTY_CLOB(),
    p_body_blob                 IN BLOB     DEFAULT EMPTY_BLOB(),
    p_parm_name                 IN apex_application_global_vc_arr2
                                DEFAULT empty_vc_arr,
    p_parm_value                IN apex_application_global_vc_arr2
                                DEFAULT empty_vc_arr,
    p_wallet_path               IN VARCHAR2 DEFAULT NULL,
    p_wallet_pwd                IN VARCHAR2 DEFAULT NULL,
    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

Parameter	Description
p_url	The URL endpoint of the Web service.
p_http_method	The HTTP method to use (PUT, POST, GET, HEAD, or DELETE).
p_username	The username if basic authentication is required for this service.

Parameter	Description
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CRED if supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_body	The HTTP payload to be sent as CLOB.
p_body_blob	The HTTP payload to be sent as binary BLOB. For example, posting a file.
p_parm_name	The name of the parameters to be used in name/value pairs.
p_parm_value	The value of the parameters to be used in name/value pairs.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.
p_credential_static_id	The name of the Web Credentials to be used. Web Credentials are configured in Workspace Utilities.
p_token_url	For token-based authentication flows (like OAuth2): The URL where to get the token from.

Example

The following example calls a RESTful-style web service using the `make_rest_request` function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared CLOB.

```

DECLARE
    l_clob      CLOB;
BEGIN

    l_clob := apex_web_service.make_rest_request(
        p_url => 'http://us.music.yahewapis.com/video/v1/list/published/
popular',
        p_http_method => 'GET',
        p_parm_name => apex_string.string_to_table('appid:format'),
        p_parm_value => apex_string.string_to_table('xyz:xml'));

END;

```

60.18 MAKE_REST_REQUEST_B Function

This function invokes a RESTful style Web service supplying either name value pairs, a character based payload, or a binary payload, and returns the response in a BLOB.

Syntax

```
APEX_WEB_SERVICE.MAKE_REST_REQUEST_B (
    p_url                      IN  VARCHAR2,
    p_http_method               IN  VARCHAR2,
    p_username                  IN  VARCHAR2 DEFAULT NULL,
    p_password                  IN  VARCHAR2 DEFAULT NULL,
    p_scheme                    IN  VARCHAR2 DEFAULT 'Basic',
    p_proxy_override             IN  VARCHAR2 DEFAULT NULL,
    p_transfer_timeout           IN  NUMBER   DEFAULT 180,
    p_body                      IN  CLOB    DEFAULT EMPTY_CLOB(),
    p_body_blob                 IN  BLOB    DEFAULT EMPTY_BLOB(),
    p_parm_name                 IN  apex_application_global_vc_arr2
                                  DEFAULT empty_vc_arr,
    p_parm_value                IN  apex_application_global_vc_arr2
                                  DEFAULT empty_vc_arr,
    p_wallet_path               IN  VARCHAR2 DEFAULT NULL,
    p_wallet_pwd                IN  VARCHAR2 DEFAULT NULL,
    p_https_host                IN  VARCHAR2 DEFAULT NULL,
    p_credential_static_id      IN  VARCHAR2 DEFAULT NULL,
    p_token_url                 IN  VARCHAR2 DEFAULT NULL )
RETURN BLOB;
```

Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_http_method	The HTTP method to use, PUT, POST, GET, HEAD, or DELETE.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_scheme	The authentication scheme, Basic (default) or AWS or Digest or OAUTH_CLIENT_CRED if supported by your database release.
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_body	The HTTP payload to be sent as CLOB.
p_body_blob	The HTTP payload to be sent as binary BLOB. For example, posting a file.
p_parm_name	The name of the parameters to be used in name/value pairs.
p_parm_value	The value of the parameters to be used in name/value pairs.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.
p_credential_static_id	The name of the Web Credentials to be used. Web Credentials are configured in Workspace Utilities.
p_token_url	For token-based authentication flows (like OAuth2): The URL where to get the token from.

Example

The following example calls a RESTful style Web service using the `make_rest_request` function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared BLOB.

```
DECLARE
    l_blob      BLOB;
BEGIN

    l_blob := apex_web_service.make_rest_request_b(
        p_url => 'http://us.music.yahooapis.com/video/v1/list/published/
popular',
        p_http_method => 'GET',
        p_parm_name => apex_string.string_to_table('appid:format'),
        p_parm_value => apex_string.string_to_table('xyz:xml'));

END;
```

60.19 OAUTH_AUTHENTICATE_CREDENTIAL Procedure

This procedure performs OAuth authentication using a credential store. The obtained access token and its expiry date are stored in a package global.

Syntax

```
APEX_WEB_SERVICE.OAUTH_AUTHENTICATE_CREDENTIAL (
    p_token_url          IN VARCHAR2,
    p_credential_static_id IN VARCHAR2,
    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

Parameter	Description
p_token_url	The url endpoint of the OAuth token service.
p_credential_static_id	The name of the Web Credentials to be used. Web Credentials are configured in Workspace Utilities.
p_proxy_override	The proxy to use for the request.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The filesystem path to a wallet if request is HTTPS. For example, file:/usr/home/oracle/WALLETS
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.

Example

```
BEGIN
    apex_web_service.oauth_authenticate_credential(
        p_token_url => '[URL to ORDS OAuth token service: http(s)://<host>:<port>/ords/.../oauth/token]',
        p_credential_static_id => '[web-credential]');
END;
```

60.20 OAUTH_AUTHENTICATE Procedure Signature 1

This procedure performs OAuth authentication and requests an OAuth access token. The token and its expiry date are stored in a package global.

Note:

Currently only the Client Credentials flow is supported.

Syntax

```
APEX_WEB_SERVICE.OAUTH_AUTHENTICATE (
    p_token_url          IN VARCHAR2,
    p_client_id           IN VARCHAR2,
    p_client_secret        IN VARCHAR2,
    p_flow_type            IN VARCHAR2 DEFAULT oauth_client_cred,
    p_proxy_override       IN VARCHAR2 DEFAULT NULL,
    p_transfer_timeout     IN NUMBER   DEFAULT 180,
    p_wallet_path          IN VARCHAR2 DEFAULT NULL,
    p_wallet_pwd           IN VARCHAR2 DEFAULT NULL,
    p_https_host            IN VARCHAR2 DEFAULT NULL,
    p_scope                IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_token_url	The URL endpoint of the OAuth token service.
p_client_id	OAuth Client ID to use for authentication.
p_client_secret	OAuth Client Secret to use for authentication.
p_flow_type	OAuth flow type. Only OAUTH_CLIENT_CRED is supported at this time.
p_proxy_override	The proxy to use for the request.
p_transfer_timeout	The amount of time in seconds to wait for a response.
p_wallet_path	The filesystem path to a wallet if request is HTTPS. For example, file:/usr/home/oracle/WALLETS
p_wallet_pwd	The password to access the wallet.
p_https_host	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.
p_scope	The OAuth scope to identify groups of attributes that will be requested from the OAuth provider. For example, profile,email

Example

```
BEGIN
    apex_web_service.oauth_authenticate(
        p_token_url      => '[URL to ORDS OAuth token service: http(s)://
{host}:{port}/ords/.../oauth/token]',
        p_client_id      => '[client-id]',
        p_client_secret  => '[client-secret]');
END;
```

60.21 OAUTH_AUTHENTICATE Procedure Signature 2 (Deprecated)

 **Note:**

OAUTH_AUTHENTICATE Procedure Signature 2 has been deprecated because `p_wallet_path` and `p_wallet_pwd` do not have a default value. Oracle recommends using `OAUTH_AUTHENTICATE_CREDENTIAL` instead.

This procedure performs OAuth authentication and requests an OAuth access token. The obtained access token and its expiry date are stored in a package global.

Syntax

```
APEX_WEB_SERVICE.OAUTH_AUTHENTICATE(
    p_token_url          IN VARCHAR2,
    p_credential_static_id IN VARCHAR2,
    p_proxy_override     IN VARCHAR2 DEFAULT NULL,
    p_transfer_timeout   IN NUMBER    DEFAULT 180,
    p_wallet_path        IN VARCHAR2,
    p_wallet_pwd         IN VARCHAR2,
    p_https_host         IN VARCHAR2 DEFAULT NULL);
```

Parameters

Parameter	Description
<code>p_token_url</code>	The url endpoint of the OAuth token service.
<code>p_credential_static_id</code>	The name of the Web Credentials to be used. Web Credentials are configured in Workspace Utilities.
<code>p_proxy_override</code>	The proxy to use for the request.
<code>p_transfer_timeout</code>	The amount of time in seconds to wait for a response.
<code>p_wallet_path</code>	The filesystem path to a wallet if request is https. For example: <code>file:/usr/home/oracle/WALLETS</code>
<code>p_wallet_pwd</code>	The password to access the wallet.
<code>p_https_host</code>	The host name to be matched against the common name (CN) of the remote server's certificate for an HTTPS request.

Example

```
BEGIN
    apex_web_service.oauth_authenticate(
        p_token_url          => '<a target="_blank"
        href="http://>http://</a>{host}:{port}/ords/scott/oauth/token',
        p_credential_static_id => 'My_credentials',
        p_wallet_path         => null,
        p_wallet_pwd          => null );
END;
```

60.22 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;
```

60.23 OAUTH_SET_TOKEN Procedure

This procedure sets the OAuth access token to be used in subsequent MAKE_REST_REQUEST calls. This procedure can be used to set a token which was obtained by different means than with OAUTH_AUTHENTICATE (such as custom code).

Syntax

```
APEX_WEB_SERVICE.OAUTH_SET_TOKEN (
    p_token    IN VARCHAR2,
    p_expires  IN DATE DEFAULT NULL );
```

Parameters

Parameter	Description
p_token	The OAuth access token to be used by MAKE_REST_REQUEST calls.
p_expires	(Optional) The token expiry date. If NULL, no expiration date is set.

Example

```
BEGIN
    apex_web_service.oauth_set_token(
        p_token => '{oauth access token}'
    );
END;
```

60.24 PARSE_RESPONSE Function

This function parses the response from a Web service that is stored in a collection and returns the result as a VARCHAR2 type.

Syntax

```
APEX_WEB_SERVICE.PARSE_RESPONSE (
    p_collection_name  IN VARCHAR2,
    p_xpath            IN VARCHAR2,
    p_ns               IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_collection_name	The name of the collection where the Web service response is stored.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared VARCHAR2 variable.

```
declare
    l_response_msg  VARCHAR2(4000);
BEGIN
    l_response_msg := apex_web_service.parse_response(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath =>
'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/
idc:statusMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');
END;
```

60.25 PARSE_RESPONSE_CLOB Function

This function parses the response from a Web service that is stored in a collection and returns the result as a CLOB type.

Syntax

```
APEX_WEB_SERVICEPARSE_RESPONSE_CLOB (
    p_collection_name    IN VARCHAR2,
    p_xpath              IN VARCHAR2,
    p_ns                 IN VARCHAR2 DEFAULT NULL )
RETURN CLOB;
```

Parameters

Parameter	Description
p_collection_name	The name of the collection where the Web service response is stored.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared CLOB variable.

```
DECLARE
    l_response_msg  CLOB;
BEGIN
    l_response_msg := apex_web_service.parse_response_clob(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath=>
'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/
idc:statusMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn"');
END;
```

60.26 PARSE_XML Function

This function parses the response from a Web service returned as an XMLTYPE and returns the value requested as a VARCHAR2.

Syntax

```
APEX_WEB_SERVICEPARSE_XML (
    p_xml           IN XMLTYPE,
    p_xpath         IN VARCHAR2,
    p_ns            IN VARCHAR2 DEFAULT NULL )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_xml	The XML document as an XMLTYPE to parse.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example uses the `make_request` function to call a Web service and store the results in a local XMLTYPE variable. The `parse_xml` function is then used to pull out a specific node of the XML document stored in the XMLTYPE and stores it in a locally declared VARCHAR2 variable.

```
DECLARE
    l_envelope CLOB;
    l_xml XMLTYPE;
    l_movie VARCHAR2(4000);
BEGIN
    l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
        <tns:GetTheatersAndMovies>
            <tns:zipCode>43221</tns:zipCode>
            <tns:radius>5</tns:radius>
        </tns:GetTheatersAndMovies>
    </soap:Body>
</soap:Envelope>';

    l_xml := apex_web_service.make_request(
        p_url => ' http://www.ignyte.com/webservices/
ignyte.whatsshowing.webservice/moviefunctions.asmx',
        p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
        p_envelope => l_envelope );

    l_movie := apex_web_service.parse_xml(
        p_xml => l_xml,
        p_xpath => ' //GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/
Theater/Movies/Movie/Name[1]',
        p_ns => ' xmlns="http://www.ignyte.com/whatsshowing"' );
END;
```

60.27 PARSE_XML_CLOB Function

This function parses the response from a Web service returned as an XMLTYPE and returns 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 CLOB;
```

Parameters

Parameter	Description
p_xml	The XML document as an XMLTYPE to parse.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example uses the `make_request` function to call a Web service and stores the results in a local XMLTYPE variable. The `parse_xml_clob` function then fetches a specific node of the XML document that is stored in the XMLTYPE and stores it in a locally-declared CLOB 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;

```

60.28 REMOVE_REQUEST_HEADER Procedure

This procedure removes an HTTP request header (`g_request_headers`). If the header parameter name does not exist, no error is raised.

Caution:

This procedure may reorder the header entries in `g_request_headers`.

Syntax

```
APEX_WEB_SERVICE.REMOVE_REQUEST_HEADER (
    p_name    IN VARCHAR2 )
```

Parameters

Parameter	Description
<code>p_name</code>	The name of the header to remove.

Example

The following example removes the "Metadata-Context" header.

```
BEGIN
    apex_web_service.remove_request_header(
        p_name => 'Metadata-Context');
END;
```

60.29 SET_REQUEST_ECID_CONTEXT Procedure

This procedure adds an Execution Context ID (ECID) HTTP request header to `g_request_headers`. This overrides the application level security setting "Pass ECID."

Syntax

```
APEX_WEB_SERVICE.SET_REQUEST_ECID_CONTEXT (
    p_ecid    IN VARCHAR2 DEFAULT AUTO_ECID )
```

Parameters

Parameter	Description
<code>p_ecid</code>	The identifier for the execution context. Options include: <ul style="list-style-type: none"> • <code>AUTO_ECID</code> - (Default) An automatically determined ECID header is added. • <code>NULL</code> - No ECID header is added. • If neither, <code>substrb(p_ecid, 1, 64)</code> is used as the ECID header.

Example 1

On application level, the sending of an ECID header is switched off. By calling `set_request_ecid_context` without any parameter, the following call to `make_rest_request` has an "ECID-Context" request header with an automatically determined ECID.

```
BEGIN
    apex_web_service.set_request_ecid_context;
END;
```

Example 2

On application level, the sending of an ECID header is switched off. By providing an ECID value, the following call to `make_rest_request` has an "ECID-Context" request header.

```
BEGIN
    apex_web_service.set_request_ecid_context(
        p_ecid => 'my-123456' );
END;
```

Example 3

On application level, the sending of an ECID header is switched on. By providing an ECID value of `NULL`, the following call to `make_rest_request` has **no** "ECID-Context" request header.

```
BEGIN
    apex_web_service.set_request_ecid_context(
        p_ecid => null );
END;
```

Example 4

On application level, the sending of an ECID header is switched on. By providing an ECID value, the following call to `make_rest_request` uses this value instead of an automatically determined ECID.

```
BEGIN
    apex_web_service.set_request_ecid_context(
        p_ecid => 'my-123456' );
END;
```

60.30 SET_REQUEST_HEADERS Procedure

This procedure sets HTTP request headers (`g_request_headers`) for subsequent `MAKE_REQUEST` or `MAKE_REST_REQUEST` calls.

Syntax

```
APEX_WEB_SERVICE.SET_REQUEST_HEADERS (
    p_name_01          IN VARCHAR2,
    p_value_01          IN VARCHAR2,
    p_name_02          IN VARCHAR2 DEFAULT NULL,
```

```

    p_value_02      IN VARCHAR2 DEFAULT NULL,
    p_name_03      IN VARCHAR2 DEFAULT NULL,
    p_value_03      IN VARCHAR2 DEFAULT NULL,
    p_name_04      IN VARCHAR2 DEFAULT NULL,
    p_value_04      IN VARCHAR2 DEFAULT NULL,
    p_name_05      IN VARCHAR2 DEFAULT NULL,
    p_value_05      IN VARCHAR2 DEFAULT NULL,
    p_reset        IN BOOLEAN  DEFAULT TRUE,
    p_skip_if_exists IN BOOLEAN DEFAULT FALSE );

```

Parameters

Parameter	Description
p_name_01	Name of the 1st parameter to set.
p_value_01	Value of the 1st parameter to set.
p_name_02	Name of the 2nd parameter to set.
p_value_02	Value of the 2nd parameter to set.
p_name_03	Name of the 3rd parameter to set.
p_value_03	Value of the 3rd parameter to set.
p_name_04	Name of the 4th parameter to set.
p_value_04	Value of the 4th parameter to set.
p_name_05	Name of the 5th parameter to set.
p_value_05	Value of the 5th parameter to set.
p_reset	Whether to clear the request header array before.
p_skip_if_exists	If TRUE, any existing headers with the same name remain unchanged. For example, if you pass in "Content-Type" as p_name_NN and that header is already present in the apex_web_services.g_request_headers array, then the value that you pass in does not override the existing header value for that name.

Example 1

The following example appends "Content-Type" and "User-Agent" HTTP request headers to the already existing headers, but only if they do not exist yet.

```

begin
    apex_web_service.set_request_headers(
        p_name_01      => 'Content-Type',
        p_value_01     => 'application/json',
        p_name_02      => 'User-Agent',
        p_value_02     => 'APEX',
        p_reset        => false,
        p_skip_if_exists => true );
end;

```

Example 2

The following example clears existing request headers and sets "Content-Type" and "User-Agent."

```

begin
    apex_web_service.set_request_headers(

```

```
p_name_01      => 'Content-Type',
p_value_01     => 'application/json',
p_name_02      => 'User-Agent',
p_value_02     => 'APEX' );
end;
```

61

APEX_WORKFLOW

The APEX_WORKFLOW package provides API's for the management of Workflows in Oracle APEX. This package is part of the APEX Workflow functionality.

- Constants and Data Types
- [CONTINUE_ACTIVITY Procedure Signature 1](#)
- [CONTINUE_ACTIVITY Procedure Signature 2](#)
- [GET_LOV_ACTIVITY_STATE Function](#)
- [GET_LOV_WORKFLOW_STATE Function](#)
- [GET_NEXT_PURGE_TIMESTAMP Function](#)
- [GET_VARIABLE_CLOB_VALUE Function](#)
- [GET_VARIABLE_VALUE Function](#)
- [GET_WORKFLOW_STATE Function](#)
- [GET_WORKFLOWS Function](#)
- [IS_ADMIN Function](#)
- [IS_ALLOWED Function](#)
- [IS_OF_PARTICIPANT_TYPE Function](#)
- [REFRESH_PARTICIPANTS Procedure](#)
- [REMOVE_DEVELOPMENT_INSTANCES Procedure](#)
- [RESUME Procedure](#)
- [RETRY Procedure](#)
- [SET_LOG_LEVEL Procedure](#)
- [START_WORKFLOW Function](#)
- [SUSPEND Procedure](#)
- [TERMINATE Procedure](#)
- [TERMINATEFAULTEDWORKFLOWS Procedure](#)
- [UPDATE_VARIABLES Procedure](#)

61.1 Constants and Data Types

Constants

The APEX_WORKFLOW package uses the following constants.

```
c_workflow_system_user  constant varchar2(8)  := 'system';
```

Workflow and Activity (Instance) States

```
c_state_active      constant t_workflow_state := 'ACTIVE';
c_state_terminated constant t_workflow_state := 'TERMINATED';
c_state_completed  constant t_workflow_state := 'COMPLETED';
c_state_faulted   constant t_workflow_state := 'FAULTED';
c_state_suspended constant t_workflow_state := 'SUSPENDED';
c_state_waiting    constant t_workflow_state := 'WAITING';
```

Workflow (Instance) Operations

```
c_workflow$op_suspend      constant t_workflow_operation := 'SUSPEND';
c_workflow$op_resume       constant t_workflow_operation := 'RESUME';
c_workflow$op_retry        constant t_workflow_operation := 'RETRY';
c_workflow$op_update_var  constant t_workflow_operation := 'UPDATE_VARIABLE';
c_workflow$op_terminate    constant t_workflow_operation := 'TERMINATE';
```

Workflow Substitution Strings

```
c_workflow_id           constant varchar2(30)      := 'APEX$WORKFLOW_ID';
c_workflow_activity_id  constant varchar2(30)      := 'APEX$WORKFLOW_ACTIVITY_ID';
c_workflow_initiator    constant varchar2(30)      := 'APEX$WORKFLOW_INITIATOR';
c_workflow_state         constant varchar2(30)      := 'APEX$WORKFLOW_STATE';
c_workflow_detail_pk    constant varchar2(30)      := 'APEX$WORKFLOW_DETAIL_PK';
c_workflow_created_on   constant varchar2(30)      := 'APEX$WORKFLOW_CREATED_ON';
```

Workflow Activity (Instance) Status

```
c_activity_status_success constant t_activity_status := 'SUCCESS';
c_activity_status_failure  constant t_activity_status := 'FAILURE';
```

Workflow Parameters Default

```
c_empty_workflow_parameters t_workflow_parameters;
```

Workflow Participant Types

```
c_workflow_owner          constant t_workflow_participant_type := 'OWNER';
c_workflow_admin          constant t_workflow_participant_type := 'ADMIN';
```

Workflow List Context Types

```
c_context_my_workflows   constant t_workflow_list_context := 'MY_WORKFLOWS';
c_context_admin_workflows constant t_workflow_list_context := 'ADMIN_WORKFLOWS';
```

```
c_context_initiated_by_me      constant t_workflow_list_context :=  
'INITIATED_BY_ME';  
c_context_single_workflow     constant t_workflow_list_context :=  
'SINGLE_WORKFLOW';
```

Data Types

The APEX_WORKFLOW package uses the following data types.

Global Data Types

```
subtype t_workflow_state          is varchar2(10);  
subtype t_activity_status        is varchar2(15);  
subtype t_workflow_participant_type is varchar2(15);  
subtype t_workflow_list_context   is varchar2(15);  
subtype t_workflow_operation      is varchar2(30);
```

Workflow Parameters (Value)

Value	Description
static_id	The static ID of the parameter. This ID must match the static ID of the corresponding parameter in the workflow definition.
value	The value of the parameter as a session state value.
string_value	(Deprecated) The value of the parameter as a string.
format_mask	(Optional) Format mask for the parameter.

```
type t_workflow_parameter is record (
    static_id      varchar2(255),
    value          apex_session_state.t_value,
    string_value   varchar2(32767),
    format_mask    varchar2(255));
```

Collection of Workflow Parameter Values

```
type t_workflow_parameters  is table of t_workflow_parameter index by
pls_integer;
```

Collection of Workflow Participant Types

```
type t_workflow_participant_types is table of t_workflow_participant_type
index by pls_integer;
```

61.2 CONTINUE_ACTIVITY Procedure Signature 1

This procedure continues the Workflow at the given activity. The workflow activity must be in the "WAITING" state.

Syntax

```
APEX_WORKFLOW.CONTINUE_ACTIVITY (
    p_instance_id          IN NUMBER,
    p_static_id             IN VARCHAR2,
    p_activity_params       IN wwv_flow_global_vc_map,
    p_activity_status       IN t_activity_status DEFAULT
                            c_activity_status_success );
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_static_id	Static ID of the activity.
p_activity_params	The parameters returned as part of the activity execution. If these parameters correspond to workflow variables, the values of those variables will get updated with what is provided here, before the workflow continues to the next activity.
p_activity_status	The status of the activity. Default SUCCESS.

Example

The following example continues a Workflow activity.

```
DECLARE
    l_activity_params apex_global_vc_map;
BEGIN
    l_activity_result('TASK_OUTCOME') := 'APPROVED';
    apex_workflow.continue_activity(
        p_instance_id      => 1234,
        p_static_id        => 'REQUEST_LEAVE_APPROVAL',
        p_activity_params  => l_activity_params);
END;
```

61.3 CONTINUE_ACTIVITY Procedure Signature 2

This procedure continues the Workflow at the given activity. The activity must be in "WAITING" state.

Syntax

```
APEX_WORKFLOW.CONTINUE_ACTIVITY (
    p_instance_id          IN NUMBER,
    p_activity_instance_id IN NUMBER,
    p_activity_params       IN wwv_flow_global_vc_map,
    p_activity_status       IN t_activity_status DEFAULT
                            c_activity_status_success);
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_activity_instance_id	ID of the activity instance.
p_activity_params	The parameters returned as part of the activity execution. If these parameters correspond to Workflow variables, the values of those variables are updated with what is provided here before the workflow continues to the next activity.
p_activity_status	The status of the activity. Default SUCCESS.

Example

The following example continues a Workflow Activity.

```

DECLARE
    l_activity_params apex_global_vc_map;
BEGIN
    l_activity_result('TASK_OUTCOME') := 'APPROVED';
    apex_workflow.continue_activity(
        p_instance_id      => 1234,
        p_activity_instance_id => 1,
        p_activity_params     => l_activity_params);
END;

```

61.4 GET_LOV_ACTIVITY_STATE Function

This function gets the list of value data for the activity instance attribute state.

Syntax

```

APEX_WORKFLOW.GET_LOV_ACTIVITY_STATE
RETURN apex_t_temp_lov_data;

```

Parameters

None.

Returns

A table of apex_t_temp_lov_data.

Example

```

select disp,
       val
  from table ( apex_workflow.get_lov_activity_state )

```

61.5 GET_LOV_WORKFLOW_STATE Function

This function gets the list of value data for the workflow instance attribute state.

Syntax

```
APEX_WORKFLOW.GET_LOV_WORKFLOW_STATE
RETURN apex_t_temp_lov_data;
```

Parameters

None.

Returns

A table of apex_t_temp_lov_data.

Example

```
select disp,
       val
  from table ( apex_workflow.get_lov_workflow_state )
```

61.6 GET_NEXT_PURGE_TIMESTAMP Function

This function retrieves the timestamp of the next purge.

Syntax

```
APEX_WORKFLOW.GET_NEXT_PURGE_TIMESTAMP
RETURN timestamp with time zone;
```

Parameters

None.

Returns

Returns the timestamp of the next purge.

Example

```
DECLARE
    l_next_purge_job_ts timestamp with time zone;
BEGIN
    l_next_purge_job_ts := apex_workflow.get_next_purge_timestamp();
END;
```

61.7 GET_VARIABLE_CLOB_VALUE Function

This function gets the CLOB value of a workflow variable. It returns the VARCHAR2 value if the data type of the variable is VARCHAR2. It returns NULL if the variable is not of CLOB or VARCHAR2 datatype.

Syntax

```
APEX_WORKFLOW.GET_VARIABLE_CLOB_VALUE (
    p_instance_id          IN NUMBER,
    p_variable_static_id    IN VARCHAR2 )
RETURN CLOB;
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_variable_static_id	Static ID of the variable.

Returns

CLOB

Example

The following example returns the value of the Workflow variable called "REST_RESPONSE."

```
BEGIN
    apex_workflow.get_variable_clob_value(
        p_instance_id => 1234,
        p_variable_static_id => 'REST_RESPONSE');
END;
```

61.8 GET_VARIABLE_VALUE Function

This function gets the string value of a workflow variable.

If the workflow variable has a format mask set, the same format mask is applied to the value being returned.

If the workflow variable is of CLOB data type and the data length is greater than VARCHAR2 max length then an exception is thrown if `p_raise` is set to TRUE.

Syntax

```
APEX_WORKFLOW.GET_VARIABLE_VALUE (
    p_instance_id          IN NUMBER,
    p_variable_static_id    IN VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_variable_static_id	Static ID of the variable.

Returns

VARCHAR2.

Example

The following example returns the value of Workflow Variable called "NEW_SALARY."

```
BEGIN
    apex_workflow.get_variable_value(
        p_instance_id => 1234,
        p_variable_static_id => 'NEW_SALARY');
END;
```

61.9 GET_WORKFLOW_STATE Function

This function gets the current state of the workflow.

Syntax

```
APEX_WORKFLOW.GET_WORKFLOW_STATE (
    p_instance_id           IN NUMBER )
RETURN t_workflow_state;
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.

Returns

t_workflow_state

Example

The following example gets the current state of the Workflow Instance.

```
BEGIN
    return apex_workflow.get_workflow_state(
        p_instance_id => 1234);
END;
```

61.10 GET_WORKFLOWS Function

This function gets the workflows of a user depending on the given context. Context can be MY_WORKFLOWS, ADMIN_WORKFLOWS, or SINGLE_WORKFLOW.

 **Note:**

The function only returns data in the context of a valid Oracle APEX session. It returns no data in SQL Workshop.

Syntax

```
APEX_WORKFLOW.GET_WORKFLOWS (
    p_context          IN VARCHAR2 DEFAULT
                        apex_workflow_api.c_context_my_workflows,
    p_user             IN VARCHAR2 DEFAULT apex_security.g_user,
    p_workflow_id      IN NUMBER   DEFAULT NULL,
    p_application_id   IN NUMBER   DEFAULT NULL )
RETURN apex_t_workflow_instances pipelined;
```

Parameters

Parameter	Description
p_context	The list context. Default is MY_WORKFLOWS.
p_user	The user to check for. Default is logged-in user. Needs p_context set to MY_WORKFLOWS, ADMIN_WORKFLOWS or INITIATED_BY_ME.
p_workflow_id	Filter for a workflow ID instead of a user. Default is NULL. Needs p_context set to SINGLE_WORKFLOW.
p_application_id	Filter for an application. Default is NULL (all applications).

Returns

A table of workflows (type apex_t_workflow_instances) containing the following columns:

- badge_css_classes varchar2(255)
- badge_state varchar2(255)
- badge_text varchar2(255)
- created_ago varchar2(255)
- created_ago_hours number
- created_by varchar2(255)
- created_on timestamp with time zone
- duration_seconds number
- end_time timestamp with time zone
- initiator varchar2(255)
- initiator_lower varchar2(255)

- is_completed varchar2(1)
- is_dev_mode varchar2(1)
- is_terminated varchar2(1)
- last_updated_by varchar2(255)
- last_updated_on timestamp with time zone
- parent_activity_id number
- parent_id number
- start_time timestamp with time zone
- state varchar2(255)
- state_code varchar2(32)
- title varchar2(4000)
- workflow_def_app_id number
- workflow_def_app_name varchar2(255)
- workflow_def_id number
- workflow_def_name varchar2(255)
- workflow_def_static_id varchar2(255)
- workflow_id number
- workflow_version varchar2(255)
- workflow_version_id number

Example

```
select *
  from table ( apex_workflow.get_workflows (
    p_context => 'MY_WORKFLOWS' ) )
```

61.11 IS_ADMIN Function

This function checks whether the given user is a business administrator for at least one workflow in this application.

Syntax

```
APEX_WORKFLOW.IS_ADMIN (
    p_user          IN VARCHAR2 DEFAULT wvv_flow_security.g_user,
    p_application_id IN NUMBER   DEFAULT NULL )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_user	The user to check for. Default is logged-in user.
p_application_id	The application to check for. Default behavior checks against all applications in the workspace.

Returns

TRUE if the user given by `p_user` is defined as a Business Admin for at least one workflow in the given application. Otherwise FALSE.

Example

```
DECLARE
    l_is_business_admin boolean;
BEGIN
    l_is_admin := apex_workflow.is_admin(
        p_user => 'STIGER'
    );
    IF l_is_admin THEN
        dbms_output.put_line('STIGER is a Business Administrator');
    END IF;
END;
```

61.12 IS_ALLOWED Function

This function checks whether the given user is allowed to perform a certain operation on a Workflow.

Syntax

```
APEX_WORKFLOW.IS_ALLOWED (
    p_instance_id           IN NUMBER,
    p_operation              IN wwv_flow_workflow_api.t_workflow_operation,
    p_user                   IN VARCHAR2 DEFAULT wwv_flow_security.g_user )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
<code>p_instance_id</code>	The Workflow ID.
<code>p_operation</code>	The operation to check.
<code>p_user</code>	The user to check for. Default is logged-in user.

Returns

TRUE if the user given by `p_user` is allowed to perform the operation given by `p_operation`. Otherwise FALSE.

Example

```
DECLARE
    l_is_allowed boolean;
BEGIN
    l_is_allowed := apex_workflow.is_allowed(
        p_instance_id      => 1234,
        p_operation        => apex_workflow.c_workflow_op_suspend,
        p_user             => 'STIGER'
```

```

);
IF l_is_allowed THEN
    dbms_output.put_line('STIGER is allowed to suspend the workflow
1234');
END IF;
END;

```

61.13 IS_OF_PARTICIPANT_TYPE Function

This function checks whether the given user is of a certain participant type for a Workflow.

Syntax

```
APEX_WORKFLOW.IS_OF_PARTICIPANT_TYPE (
    p_instance_id           IN NUMBER,
    p_participant_type      IN t_workflow_participant_type,
    p_user                   IN VARCHAR2 )
RETURN BOOLEAN;
```

Parameters

Parameter	Description
p_instance_id	The Workflow ID.
p_participant_type	The participant type.
p_user	The user to check for.

Returns

TRUE if the user given by p_user is a participant of given participant type for a given workflow. Otherwise FALSE.

Example

The following example checks if if the user STIGER is of participant type OWNER for Workflow 1234.

```

DECLARE
    l_is_owner boolean;
BEGIN
    l_is_owner := apex_workflow.is_of_participant_type(
        p_instance_id    => 1234,
        p_participant_type => apex_workflow.c_workflow_owner,
        p_user           => 'STIGER'
    );
    IF l_is_owner THEN
        dbms_output.put_line('STIGER is an owner for workflow 1234');
    END IF;
END;
```

61.14 REFRESH_PARTICIPANTS Procedure

This procedure refreshes the participants for this workflow instance.

Syntax

```
APEX_WORKFLOW.REFRESH_PARTICIPANTS (
    p_instance_id    IN NUMBER );
```

Parameters

Parameter	Description
p_instance_id	The Workflow ID.

Example

```
BEGIN
    apex_workflow.refresh_participants(
        p_instance_id      => 1234
    );
END;
```

61.15 REMOVE_DEVELOPMENT_INSTANCES Procedure

This procedure removes workflow instances created in "DEV" mode. This API can be used by a developer to remove any workflow instances that have been started from App Builder in DEV mode.

Syntax

```
APEX_WORKFLOW.REMOVE DEVELOPMENT_INSTANCES (
    p_application_id   IN NUMBER    DEFAULT apex_application.g_flow_id,
    p_static_id        IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_application_id	The application ID.
p_static_id	Static ID of the workflow. If omitted, removes all development workflow instances of the application.

Example

```
BEGIN
    apex_workflow.remove_development_instances(
        p_application_id => 100);
END;
```

61.16 RESUME Procedure

This procedure resumes a Workflow. Only suspended Workflow Instances can be resumed.

If the application is running in a runtime-only session, only Workflow Administrators can perform this operation.

Syntax

```
APEX_WORKFLOW.RESUME (
    p_instance_id          IN NUMBER,
    p_activity_static_id   IN VARCHAR2 DEFAULT NULL );
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_activity_static_id	Static ID of the Activity to resume the Workflow or NULL if to resume at current activity.

Example

Below example resumes a Workflow Instance at activity "UPDATE_JOB_LEVEL." Only suspended Workflow Instances can be resumed.

```
BEGIN
    apex_workflow.suspend(
        p_instance_id      => 1234,
        p_activity_static_id => 'UPDATE_JOB_LEVEL');
END;
```

61.17 RETRY Procedure

This procedure retries a Workflow.

Syntax

```
APEX_WORKFLOW.RETRY (
    p_instance_id          IN NUMBER );
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.

Example

The following example retries a faulted Workflow Instance. The activity at which the fault was encountered is the point where the API retries execution.

```
BEGIN
    apex_workflow.retry(
        p_instance_id => 1234);
END;
```

61.18 SET_LOG_LEVEL Procedure

This procedure sets the debug log level for the Workflow instance. When set, this overrides the debug log level settings for the Workflow version.

Available values:

- 1 - apex_debug_api.c_log_level_error
- 2 - apex_debug_api.c_log_level_warn
- 4 - apex_debug_api.c_log_level_info
- 5 - apex_debug_api.c_log_level_enter
- 6 - apex_debug_api.c_log_level_app_trace

Syntax

```
APEX_WORKFLOW.SET_LOG_LEVEL (
    p_instance_id      IN NUMBER,
    p_debug_level      IN apex_debug_api.t_log_level );
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_debug_level	The debug level to use.

Example

The following example sets the debug level of a Workflow instance.

```
BEGIN
    apex_workflow.set_log_level (
        p_instance_id    => 1,
        p_debug_level     => apex_debug_api.c_log_level_app_trace);
END;
```

61.19 START_WORKFLOW Function

This function starts a new Workflow given the Workflow definition ID.

Syntax

```
APEX_WORKFLOW.START_WORKFLOW (
    p_application_id      IN NUMBER
                                DEFAULT apex_application.g_flow_id,
    p_static_id           IN VARCHAR2,
    p_parameters          IN t_workflow_parameters
                                DEFAULT c_empty_workflow_parameters,
    p_initiator            IN VARCHAR2
                                DEFAULT NULL,
    p_detail_pk            IN VARCHAR2
                                DEFAULT NULL,
```

```

    p_debug_level           IN apex_debug_api.t_log_level  DEFAULT NULL )
RETURN NUMBER;

```

Parameters

Parameter	Description
p_application_id	The application ID that creates the Workflow.
p_static_id	Static ID of the Workflow definition.
p_parameters	Optional workflow parameters.
p_initiator	(Optional) Initiator information for the workflow.
p_detail_pk	(Optional) Detail Primary Key.
p_debug_level	(Optional) Debug log level for the Workflow instance being started.

Returns

The ID of the newly started workflow.

Example

The following example starts a Workflow for a given requisition.

```

BEGIN
    l_workflow_id := apex_workflow.start_workflow (
        p_application_id => 110,
        p_static_id      => 'REQUISITIONWORKFLOW',
        p_parameters     => apex_workflow.t_workflow_parameters(
            1 => apex_workflow.t_workflow_parameter(static_id => 'REQ_DATE',
string_value   => sysdate),
            3 => apex_workflow.t_workflow_parameter(static_id =>
'REQ_AMOUNT', string_value => l_req_amount),
            4 => apex_workflow.t_workflow_parameter(static_id => 'REQ_ITEM',
string_value   => l_req_item),
            5 => apex_workflow.t_workflow_parameter(static_id => 'REQ_ID',
string_value   => l_req_id)),
        p_debug_level => apex_debug_api.c_log_level_info );
END;

```

61.20 SUSPEND Procedure

This procedure suspends a Workflow. Only Active or Faulted Workflow instances can be suspended.

Syntax

```

APEX_WORKFLOW.SUSPEND (
    p_instance_id          IN NUMBER );

```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.

Example

The following example suspends a Workflow instance.

```
BEGIN
    apex_workflow.suspend(
        p_instance_id => 1234);
END;
```

61.21 TERMINATE Procedure

This procedure terminates a Workflow. Only Active, Suspended, or Faulted Workflow instances can be terminated.

Syntax

```
APEX_WORKFLOW.TERMINATE (
    p_instance_id           IN NUMBER );
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.

Example

The following example terminates a Workflow Instance.

```
BEGIN
    apex_workflow.terminate(
        p_instance_id => 1234);
END;
```

61.22 TERMINATE_FAULTED_WORKFLOWS Procedure

This procedure terminates all faulted workflow instances for an application.

Syntax

```
APEX_WORKFLOW.TERMINATE_FAULTED_WORKFLOWS (
    p_application_id      IN NUMBER );
```

Parameters

Parameter	Description
p_application_id	ID of the application.

Example

The following example terminates all faulted Workflow Instances.

```
BEGIN
    apex_workflow.terminate_faulted_workflows(
        p_application_id => 110);
END;
```

61.23 UPDATE_VARIABLES Procedure

This procedure updates workflow variables of the workflow instance. If the Workflow variable has a format mask set, the same mask is applied to the value being passed here.

The logged-in user must be a Workflow Administrator to execute this API. This procedure can only run for workflows in Suspended or Faulted states.

Syntax

```
APEX_WORKFLOW.UPDATE_VARIABLES (
    p_instance_id          IN NUMBER,
    p_changed_params        IN t_workflow_parameters );
```

Parameters

Parameter	Description
p_instance_id	ID of the Workflow.
p_changed_params	Table of Workflow variables to be updated.

Example

The following example updates a Workflow variable value.

```
BEGIN
    apex_workflow.update_variables(
        p_instance_id => 1234,
        p_changed_params => apex_workflow.t_workflow_parameters(
            1 => apex_workflow.t_workflow_parameter(static_id =>
'NEW_SALARY', string_value => '2,560.50')));
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_DIR_ENTRIES Function](#)
- [GET_FILE_CONTENT Function Signature 1 \(Deprecated\)](#)
- [GET_FILE_CONTENT Function Signature 2](#)
- [GET_FILES Function \(Deprecated\)](#)

62.1 Data Types

The APEX_ZIP package uses the following data types.

t_dir_entries

An easily accessible directory of all the files in the archive that is indexed by the full name of the file including its path.

```
type t_dir_entries is table of t_dir_entry index by VARCHAR2(32767);
```

t_dir_entry

A file in the archive with precomputed metadata.

```
type t_dir_entry is record (
    file_name          VARCHAR2(32767),
    uncompressed_length NUMBER,
    is_directory       BOOLEAN );
```

Attribute	Description
file_name	Name of the compressed file including the directory path.
uncompressed_length	The size of the decompressed file in bytes.
is_directory	TRUE if the entry represents a file system directory.

t_files**Caution:**

This data type is deprecated. It will be removed in a future release. Use `t_dir_entries` instead.

Collection of file names and paths.

```
type t_files is table of varchar2(32767) index by binary_integer;
```

62.2 ADD_FILE Procedure

This procedure adds a single file to a zip file. You can call this procedure multiple times to add multiple files to the same zip file.

Tip:

After all files are added, you must call the `APEX_ZIP.FINISH` procedure.

Syntax

```
APEX_ZIP.ADD_FILE (
    p_zipped_blob IN OUT NOCOPY BLOB,
    p_file_name   IN VARCHAR2,
    p_content      IN BLOB )
```

Parameters

Parameter	Description
<code>p_zipped_blob</code>	BLOB containing the zip file.
<code>p_file_name</code>	File name, including path, of the file to be added to the zip file.
<code>p_content</code>	BLOB containing the file.

Example

This example reads multiple files from a table and puts them into a single zip file.

```
DECLARE
    l_zip_file blob;
BEGIN
    FOR l_file in ( SELECT file_name,
                           file_content
                      FROM my_files )
    LOOP
        apex_zip.add_file (
            p_zipped_blob => l_zip_file,
```

```
    p_file_name  => l_file.file_name,
    p_content     => l_file.file_content );
END LOOP;

apex_zip.finish (
    p_zipped_blob => l_zip_file );

END;
```



See Also:

[FINISH Procedure](#)

62.3 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

Parameter	Description
p_zipped_blob	BLOB containing the zip file.

Example

See [ADD_FILE Procedure](#).

62.4 GET_DIR_ENTRIES Function

This function returns a table of directory entries containing information about each file in the provided ZIP file. The returned table of records is indexed by the file names (including the path).

Syntax

```
APEX_ZIP.GET_DIR_ENTRIES (
    p_zipped_blob IN BLOB,
    p_only_files  IN BOOLEAN  DEFAULT TRUE,
    p_encoding    IN VARCHAR2 DEFAULT NULL )
RETURN t_dir_entries;
```

Parameters

Parameter	Description
p_zipped_blob	The BLOB containing the ZIP file.
p_only_files	Only return files, not directories, in the directory listing.
p_encoding	The encoding used to compress the file.

Returns

A table of directory entries.

Example

The following example reads a ZIP file from a table, extracts it, and stores all files of the ZIP file into my_files.

```

DECLARE
    l_zip_file      blob;
    l_unzipped_file blob;
    l_dir          apex_zip.t_dir_entries;
    l_file_path    varchar2(32767);
BEGIN
    SELECT file_content
        INTO l_zip_file
        FROM my_zip_files
        WHERE file_name = 'my_file.zip';

    l_dir := apex_zip.get_dir_entries (
                p_zipped_blob => l_zip_file );

    l_file_path := l_dir.first;
    WHILE l_file_path IS NOT NULL LOOP
        l_unzipped_file := apex_zip.get_file_content (
                            p_zipped_blob => l_zip_file,
                            p_dir_entry   => l_dir(l_file_path) );

        INSERT INTO my_files ( file_name, file_content )
        values ( l_file_path, l_unzipped_file );

        l_file_path := l_dir.next(l_file_path);
    END LOOP;
END;

```

See Also:

- [GET_FILE_CONTENT Function Signature 2](#)
- [GET_FILES Function \(Deprecated\)](#)

62.5 GET_FILE_CONTENT Function Signature 1 (Deprecated)

⚠ Caution:

This API is deprecated and will be removed in a future release.

Use [GET_FILE_CONTENT Function Signature 2](#) instead.

This function returns the BLOB of a file contained in a provided zip file.

Syntax

```
APEX_ZIP.GET_FILE_CONTENT (
    p_zipped_blob IN BLOB,
    p_file_name    IN VARCHAR2,
    p_encoding     IN VARCHAR2 DEFAULT NULL )
RETURN BLOB;
```

Parameters

Parameter	Description
p_zipped_blob	This is the BLOB containing the zip file.
p_file_name	File name, including path, of a file located in the zip file.
p_encoding	Encoding used to zip the file.

Returns

Return	Description
BLOB	BLOB of the file specified in p_file_name.

Example

See [GET_FILES Function \(Deprecated\)](#).

💡 See Also:

[GET_FILE_CONTENT Function Signature 2](#)

62.6 GET_FILE_CONTENT Function Signature 2

This function returns the BLOB of a file contained in a provided zip file.

Syntax

```
APEX_ZIP.GET_FILE_CONTENT (
    p_zipped_blob    IN BLOB,
    p_dir_entry      IN t_dir_entry )
RETURN BLOB;
```

Parameters

Parameter	Description
p_zipped_blob	The BLOB containing the zip file.
p_dir_entry	The directory entry describing the required file.

Returns

Return	Description
BLOB	BLOB of the file specified in the p_dir_entry record.

Example

See [GET_FILES Function \(Deprecated\)](#).



See Also:

[GET_DIR_ENTRIES Function](#)

62.7 GET_FILES Function (Deprecated)



Caution:

This API is deprecated and will be removed in a future release.

Use [GET_DIR_ENTRIES Function](#) instead.

This function returns an array of file names, including the path, of a provided zip file that contains a BLOB.

Syntax

```
APEX_ZIP.GET_FILES (
    p_zipped_blob IN BLOB,
    p_only_files  IN BOOLEAN DEFAULT TRUE,
    p_encoding    IN VARCHAR2 DEFAULT NULL )
RETURN t_files;
```

Parameters

Parameter	Description
p_zipped_blob	This is the zip file containing the BLOB.
p_only_files	If set to TRUE, empty directory entries are not included in the returned array. Otherwise, set to FALSE to include empty directory entries.
p_encoding	This is the encoding used to zip the file.

Returns

Return	Description
t_files	A table of file names and path. See " Data Types " for more details.

Example

This example demonstrates reading a zip file from a table, extracting it and storing all files of the zip file into `my_files`.

```

declare
    l_zip_file      blob;
    l_unzipped_file blob;
    l_files         apex_zip.t_files;
begin
    select file_content
        into l_zip_file
        from my_zip_files
    where file_name = 'my_file.zip';

    l_files := apex_zip.get_files (
        p_zipped_blob => l_zip_file );

    for i in 1 .. l_files.count loop
        l_unzipped_file := apex_zip.get_file_content (
            p_zipped_blob => l_zip_file,
            p_file_name   => l_files(i) );

        insert into my_files ( file_name, file_content )
        values ( l_files(i), l_unzipped_file );
    end loop;
end;

```

 **See Also:**

[GET_DIR_ENTRIES Function](#)

63

JavaScript APIs

This content has been moved to the [Oracle APEX JavaScript API Reference](#).

Index

A

ABORT procedure
 APEX_AUTOMATION, [12-1](#)
ABORT procedure signature 1
 APEX_BACKGROUND_PROCESS, [13-3](#)
ABORT procedure signature 2
 APEX_BACKGROUND_PROCESS, [13-3](#)
ADD procedure
 APEX_CSS, [17-1](#)
ADD_3RD_PARTY_LIBRARY_FILE procedure
 APEX_CSS, [17-1](#)
 APEX_JAVASCRIPT, [36-1](#)
ADD_AD_COLUMN procedure
 APEX_UI_DEFAULT_UPDATE, [58-2](#)
ADD_AD_SYNONYM procedure
 APEX_UI_DEFAULT_UPDATE, [58-3](#)
ADD_AGGREGATE procedure
 APEX_DATA_EXPORT, [20-5](#)
ADD_ATTACHMENT procedure
 APEX_MAIL, [41-2](#), [41-4](#)
ADD_ATTRIBUTE function signature 1
 APEX_JAVASCRIPT, [36-2](#)
ADD_ATTRIBUTE function signature 2
 APEX_JAVASCRIPT, [36-4](#)
ADD_ATTRIBUTE function signature 3
 APEX_JAVASCRIPT, [36-4](#)
ADD_ATTRIBUTE function signature 4
 APEX_JAVASCRIPT, [36-5](#)
ADD_AUTO_PROV_RESTRICTIONS procedure
 APEX_INSTANCE_ADMIN, [32-14](#)
ADD_BLUEPRINT procedure
 APEX_DG_DATA_GEN, [24-2](#)
ADD_BLUEPRINT_FROM_FILE procedure
 APEX_DG_DATA_GEN, [24-3](#)
ADD_BLUEPRINT_FROM_TABLES procedure
 APEX_DG_DATA_GEN, [24-4](#)
ADD_COLUMN procedure
 APEX_DATA_EXPORT, [20-7](#)
 APEX_DG_DATA_GEN, [24-5](#)
 APEX_EXEC, [27-14](#)
ADD_COLUMN_GROUP procedure
 APEX_DATA_EXPORT, [20-8](#)
ADD_DATA_SOURCE procedure
 APEX_DG_DATA_GEN, [24-8](#)

ADD_DML_ARRAY_ROW procedure
 APEX_EXEC, [27-15](#)
ADD_DML_ROW procedure
 APEX_EXEC, [27-20](#)
ADD_ERROR procedure signature 1
 APEX_ERROR, [25-5](#)
ADD_ERROR procedure signature 2
 APEX_ERROR, [25-6](#)
ADD_ERROR procedure signature 3
 APEX_ERROR, [25-7](#)
ADD_ERROR procedure signature 4
 APEX_ERROR, [25-8](#)
ADD_ERROR procedure signature 5
 APEX_ERROR, [25-9](#)
ADD_FILE procedure
 APEX_CSS, [17-2](#)
 APEX_ZIP, [62-2](#)
ADD_FILTER procedure
 APEX_EXEC, [27-21](#)
 Oracle TEXT, [27-21](#)
ADD_FILTER procedure signature 1
 APEX_IG, [33-1](#)
 APEX_IR, [34-2](#)
ADD_FILTER procedure signature 2
 APEX_IG, [33-3](#)
 APEX_IR, [34-3](#)
ADD_HIGHLIGHT procedure
 APEX_DATA_EXPORT, [20-10](#)
ADD_INLINE_CODE procedure
 APEX_JAVASCRIPT, [36-5](#)
ADD_JET procedure
 APEX_JAVASCRIPT, [36-6](#)
ADD_LIBRARY procedure
 APEX_JAVASCRIPT, [36-6](#)
ADD_MEMBER function
 APEX_COLLECTION, [15-6](#)
ADD_MEMBER procedure
 APEX_COLLECTION, [15-4](#)
ADD_MEMBERS procedure
 APEX_COLLECTION, [15-7](#)
ADD_MENU_ENTRY procedure
 APEX_EXTENSION, [29-1](#)
ADD_ONLOAD_CODE procedure
 APEX_JAVASCRIPT, [36-8](#)
ADD_ORDER_BY procedure
 APEX_EXEC, [27-27](#)

ADD_PARAMETER procedure
 APEX_EXEC, [27-28](#)
 ADD_REQUIREJS procedure
 APEX_JAVASCRIPT, [36-8](#)
 ADD_REQUIREJS_DEFINE procedure
 APEX_JAVASCRIPT, [36-8](#)
 ADD_SCHEMA procedure
 APEX_INSTANCE_ADMIN, [32-15](#)
 ADD_TABLE procedure
 APEX_DG_DATA_GEN, [24-9](#)
 ADD_TASK_COMMENT procedure
 APEX_APPROVAL, [9-4](#)
 APEX_HUMAN_TASK, [31-4](#)
 ADD_TASK_POTENTIAL_OWNER procedure
 APEX_APPROVAL, [9-5](#)
 APEX_HUMAN_TASK, [31-5](#)
 ADD_TO_HISTORY procedure
 APEX_APPROVAL, [9-6](#)
 APEX_HUMAN_TASK, [31-5](#)
 ADD_USER_ROLE procedure signature 1
 APEX_ACL, [2-1](#)
 ADD_USER_ROLE procedure signature 2
 APEX_ACL, [2-2](#)
 ADD_VALUE function signature 1
 APEX_JAVASCRIPT, [36-9](#)
 ADD_VALUE function signature 2
 APEX_JAVASCRIPT, [36-10](#)
 ADD_VALUE function signature 3
 APEX_JAVASCRIPT, [36-10](#)
 ADD_VALUE function signature 4
 APEX_JAVASCRIPT, [36-11](#)
 ADD_WEB_ENTRY_POINT procedure
 APEX_INSTANCE_ADMIN, [32-15](#)
 ADD_WORKSPACE procedure
 APEX_INSTANCE_ADMIN, [32-16](#)
 APEX_ACL, [2-1](#)
 APEX_ACL_USERS, [2-1](#)
 APEX_AI, [3-1](#)
 APEX_APP_OBJECT_DEPENDENCY, [4-1](#)
 APEX_APP_SETTING, [5-1](#)
 APEX_APPLICATION, [6-1](#)
 APEX_APPLICATION_ADMIN, [7-1](#)
 APEX_APPLICATION_INSTALL, [8-1](#)
 APEX_APPROVAL (deprecated), [9-1](#)
 APEX_AUTHENTICATION, [10-1](#)
 APEX_AUTHORIZATION, [11-1](#)
 APEX_AUTOMATION, [12-1](#)
 APEX_BACKGROUND_PROCESS, [13-1](#)
 APEX_BARCODE, [14-1](#)
 APEX_COLLECTION, [15-1](#)
 APEX_CREDENTIAL, [16-1](#)
 APEX_CSS, [17-1](#)
 APEX_CUSTOM_AUTH, [18-1](#)
 APEX_DATA_EXPORT, [20-1](#)
 APEX_DATA_INSTALL, [21-1](#)
 APEX_DATA_LOADING, [19-1](#)
 APEX_DATA_PARSER, [22-1](#)
 APEX_DEBUG, [23-1](#)
 APEX_DG_DATA_GEN, [24-1](#)
 APEX_ERROR, [25-1](#)
 APEX_ESCAPE, [26-1](#)
 APEX_EXEC, [27-1](#)
 APEX_EXPORT, [28-1](#)
 APEX_EXTENSION, [29-1](#)
 APEX_HTTP, [30-1](#)
 APEX_HUMAN_TASK, [31-1](#)
 APEX_IG, [33-1](#)
 APEX_INSTANCE_ADMIN, [32-1](#)
 APEX_IR, [34-1](#)
 APEX_ITEM, [35-1](#)
 APEX_JAVASCRIPT, [36-1](#)
 APEX_JSON, [37-1](#)
 APEX_JWT, [38-1](#)
 APEX_LANG, [39-1](#)
 APEX_LDAP, [40-1](#)
 APEX_MAIL, [41-1](#)
 APEX_MARKDOWN, [42-1](#)
 APEX_PAGE, [43-1](#)
 APEX_PLUGIN, [44-1](#)
 APEX_PLUGIN_UTIL, [45-1](#)
 APEX_PRINT, [46-1](#)
 APEX_PWA, [47-1](#)
 APEX_REGION, [48-1](#)
 APEX_REST_SOURCE_SYNC, [49-1](#)
 APEX_SEARCH, [50-1](#)
 APEX_SESSION, [51-1](#)
 APEX_SESSION_STATE, [52-1](#)
 APEX_SPATIAL, [54-1](#)
 APEX_STRING, [55-1](#)
 APEX_STRING_UTIL, [56-1](#)
 APEX_THEME, [57-1](#)
 APEX_UI_DEFAULT_UPDATE, [58-1](#)
 APEX_UTIL, [59-1](#)
 GET_EMAIL function, [59-53](#)
 IS_HIGH_CONTRAST_SESSION function, [59-84](#)
 IS_USERNAME_UNIQUE function, [59-87](#)
 REMOVE_PREFERENCE procedure, [59-96](#)
 SET_SECURITY_HIGH_CONTRAST_ON procedure, [59-120](#)
 SET_SESSION_STATE procedure, [59-123](#)
 APEX_WEB_SERVICE, [60-1](#)
 APEX_WEB_SERVICE API, [60-2](#)
 APEX_WORKFLOW, [61-1](#)
 APEX_ZIP, [62-1](#)
 APPEND_TO_MULTIPART procedure signature 1
 APEX_WEB_SERVICE, [60-8](#)
 APPEND_TO_MULTIPART procedure signature 2
 APEX_WEB_SERVICE, [60-9](#)
 application
 sending messages in APEX_MAIL_QUEUE, [41-8](#)

application (*continued*)
 sending outbound email, [41-14](#)
 sending outbound email as attachment, [41-2](#), [41-4](#)
 application installation, [8-1](#)
 APPLICATION_PAGE_ITEM_EXISTS function
 APEX_CUSTOM_AUTH, [18-1](#)
 APPLY_XLIFF_DOCUMENT procedure
 APEX_LANG, [39-1](#)
 APPROVE_TASK procedure
 APEX_APPROVAL, [9-7](#)
 APEX_HUMAN_TASK, [31-6](#)
 arrays
 APEX_APPLICATION, [6-1](#)
 G_Fnn arrays, [6-1](#)
 ASSERT_FILE_TYPE function
 APEX_DATA_PARSER, [22-2](#)
 ATTACH procedure
 APEX_SESSION, [51-1](#)
 attribute values
 setting, [59-106](#)
 AUTHENTICATE function
 APEX_LDAP, [40-1](#)
 authenticated user
 create user group, [59-19](#)
 delete user group, [59-22](#), [59-23](#)
 authentication
 scheme session cookies, [18-3](#)
 AUTO_SET_ASSOCIATED_ITEM procedure
 APEX_ERROR, [25-10](#)

B

BLOB_TO_CLOB function
 APEX_UTIL, [59-5](#)
 BLOB2CLOBBASE64 function
 APEX_WEB_SERVICE, [60-9](#)
 BUILD_REQUEST_BODY procedure
 APEX_PLUGIN_UTIL, [45-2](#)

C

CACHE_GET_DATE_OF_PAGE_CACHE
 function
 APEX_UTIL, [59-6](#), [59-7](#)
 CACHE_PURGE_BY_APPLICATION procedure
 APEX_UTIL, [59-7](#)
 CACHE_PURGE_BY_PAGE procedure
 APEX_UTIL, [59-8](#)
 CACHE_PURGE_STALE procedure
 APEX_UTIL, [59-9](#)
 Call sequence
 APEX_EXEC, [27-3](#)–[27-5](#)
 CALLBACK procedure
 APEX_AUTHENTICATION, [10-1](#)

CALLBACK2 procedure
 APEX_AUTHENTICATION, [10-4](#)
 CANCEL_TASK procedure
 APEX_APPROVAL, [9-8](#)
 APEX_HUMAN_TASK, [31-7](#)
 CHANGE_CURRENT_USER_PW procedure
 APEX_UTIL, [59-9](#)
 CHANGE_GEOM_METADATA procedure
 APEX_SPATIAL, [54-2](#)
 CHANGE_PASSWORD_ON_FIRST_USE
 function
 APEX_UTIL, [59-10](#)
 CHANGE_REPORT_OWNER procedure
 APEX_IG, [33-5](#)
 APEX_IR, [34-5](#)
 CHANGE_SUBSCRIPTION_EMAIL procedure
 APEX_IR, [34-5](#)
 CHANGE_SUBSCRIPTION_LANG procedure
 APEX_IR, [34-6](#)
 CHAT function signature 1
 APEX_AI, [3-1](#)
 CHAT function signature 2
 APEX_AI, [3-3](#)
 CHECKBOX2 function
 APEX_ITEM, [35-1](#)
 CIRCLE_POLYGON function
 APEX_SPATIAL, [54-2](#)
 CLAIM_TASK procedure
 APEX_APPROVAL, [9-9](#)
 APEX_HUMAN_TASK, [31-7](#)
 CLEAR procedure
 APEX_REGION, [48-1](#)
 CLEAR_ALL procedure
 APEX_APPLICATION_INSTALL, [8-6](#)
 CLEAR_ALL_USERS_STYLE procedure
 APEX_THEME, [57-1](#)
 CLEAR_APP_CACHE procedure
 APEX_UTIL, [59-13](#)
 CLEAR_CACHE procedure
 APEX_APP_OBJECT_DEPENDENCY, [4-1](#)
 CLEAR_COMPONENT_VALUES procedure
 APEX_PLUGIN_UTIL, [45-4](#)
 CLEAR_DML_ROWS procedure
 APEX_EXEC, [27-31](#)
 CLEAR_PAGE_CACHE procedure
 APEX_UTIL, [59-13](#)
 CLEAR_REPORT procedure signature 1
 APEX_IG, [33-5](#)
 APEX_IR, [34-7](#)
 CLEAR_REPORT procedure signature 2
 APEX_IG, [33-6](#)
 APEX_IR, [34-7](#)
 CLEAR_REQUEST_COOKIES procedure
 APEX_WEB_SERVICE, [60-10](#)
 CLEAR_REQUEST_HEADERS procedure
 APEX_WEB_SERVICE, [60-11](#)

CLEAR_TOKENS procedure
 APEX_CREDENTIAL, 16-1

CLEAR_USER_CACHE procedure
 APEX_UTIL, 59-14

CLEAR_USER_STYLE procedure
 APEX_THEME, 57-2

clearing session state
 collections, 15-4

clicks, counting, 59-14

CLOB_TO_BLOB function
 APEX_UTIL, 59-11

CLOBBASE642BLOB function
 APEX_WEB_SERVICE, 60-11

CLONE_REPORT function
 APEX_IR, 34-8

CLOSE procedure
 APEX_EXEC, 27-31

CLOSE_ALL procedure
 APEX_JSON, 37-4

CLOSE_ARRAY procedure
 APEX_EXEC, 27-31
 APEX_JSON, 37-5

CLOSE_OBJECT procedure
 APEX_JSON, 37-5

CLOSE_OPEN_DB_LINKS procedure
 APEX_UTIL, 59-12

COLLECTION_EXISTS function
 APEX_COLLECTION, 15-9

COLLECTION_HAS_CHANGED function
 APEX_COLLECTION, 15-9

COLLECTION_MEMBER_COUNT function
 APEX_COLLECTION, 15-10

collections
 accessing, 15-2
 APEX_COLLECTION API, 15-2
 clearing session state, 15-4
 determining status, 15-3

COLUMN_EXISTS function
 APEX_EXEC, 27-32

COMPLETE_TASK procedure
 APEX_APPROVAL, 9-9
 APEX_HUMAN_TASK, 31-8

constants
 APEX_AI, 3-1
 APEX_APP_OBJECT_DEPENDENCY, 4-1
 APEX_APPLICATION, 6-3
 APEX_APPLICATION_ADMIN, 7-2
 APEX_APPROVAL, 9-2
 APEX_AUTHENTICATION, 10-1
 APEX_BACKGROUND_PROCESS, 13-1
 APEX_DATA_EXPORT, 20-1
 APEX_DATA_PARSER, 22-1
 APEX_DEBUG, 23-2
 APEX_ESCAPE, 26-2
 APEX_EXEC, 27-6
 APEX_HUMAN_TASK, 31-2

constants (*continued*)
 APEX_IR, 34-1
 APEX_JSON, 37-3
 APEX_MARKDOWN, 42-1
 APEX_PAGE, 43-1
 APEX_PLUGIN, 44-2
 APEX_PRINT, 46-1
 APEX_SESSION_STATE, 52-1
 APEX_SHARED_COMPONENT, 53-1
 APEX_WEB_SERVICE, 60-7
 APEX_WORKFLOW, 61-1

CONTINUE_ACTIVITY procedure signature 1
 APEX_WORKFLOW, 61-3

CONTINUE_ACTIVITY procedure signature 2
 APEX_WORKFLOW, 61-4

COPY_DATA procedure
 APEX_EXEC, 27-33

COUNT_CLICK procedure
 APEX_UTIL, 59-14

CREATE_CLOUD_CREDENTIAL procedure
 APEX_INSTANCE_ADMIN, 32-17

CREATE_COLLECTION procedure
 APEX_COLLECTION, 15-10

CREATE_COLLECTION_FROM_QUERY procedure
 APEX_COLLECTION, 15-11

CREATE_COLLECTION_FROM_QUERY_B procedure
 APEX_COLLECTION, 15-13

CREATE_COLLECTION_FROM_QUERY_B procedure (no bind version)
 APEX_COLLECTION, 15-15

CREATE_COLLECTION_FROM_QUERY2 procedure
 APEX_COLLECTION, 15-12

CREATE_COLLECTION_FROM_QUERYB2 procedure
 APEX_COLLECTION, 15-16

CREATE_COLLECTION_FROM_QUERYB2 procedure (no bind version)
 APEX_COLLECTION, 15-17

CREATE_CREDENTIAL procedure signature 1
 APEX_CREDENTIAL, 16-2

CREATE_CREDENTIAL procedure signature 2
 APEX_CREDENTIAL, 16-3

CREATE_LANGUAGE_MAPPING procedure
 APEX_LANG, 39-2

CREATE_MESSAGE procedure
 APEX_LANG, 39-3

CREATE_OR_TRUNCATE_COLLECTION procedure
 APEX_COLLECTION, 15-18

CREATE_OR_UPDATE_ADMIN_USER procedure
 APEX_INSTANCE_ADMIN, 32-18

CREATE_SCHEMA_EXCEPTION procedure
 APEX_INSTANCE_ADMIN, [32-18](#)
 CREATE_SESSION procedure
 APEX_SESSION, [51-2](#)
 CREATE_TASK function
 APEX_APPROVAL, [9-10](#)
 APEX_HUMAN_TASK, [31-9](#)
 CREATE_USER procedure
 APEX_UTIL, [59-15](#)
 CREATE_USER_GROUP procedure
 APEX_UTIL, [59-19](#)
 CSS_SELECTOR function
 APEX_ESCAPE, [26-2](#)
 CSV function
 APEX_ESCAPE, [26-2](#)
 CSV function signature 2
 APEX_ESCAPE, [26-3](#)
 CURRENT_PAGE_IS_PUBLIC function
 APEX_CUSTOM_AUTH, [18-2](#)
 CURRENT_ROW_CHANGED function
 API_PLUGIN_UTIL, [45-4](#)
 CURRENT_USER_IN_GROUP function
 APEX_UTIL, [59-20](#)
 CUSTOM_CALENDAR procedure
 APEX_UTIL, [59-21](#)

D

data types
 APEX_AI, [3-1](#)
 APEX_APPLICATION_ADMIN, [7-2](#)
 APEX_APPROVAL, [9-2](#)
 APEX_BACKGROUND_PROCESS, [13-2](#)
 APEX_DATA_EXPORT, [20-3](#)
 APEX_DATA_LOADING, [19-1](#)
 APEX_DATA_PARSER, [22-1](#)
 APEX_EXEC, [27-10](#)
 APEX_HUMAN_TASK, [31-2](#)
 APEX_IR, [34-1](#)
 APEX_JSON, [37-3](#)
 APEX_PLUGIN, [44-3](#)
 APEX_SESSION_STATE, [52-1](#)
 APEX_SPATIAL, [54-1](#)
 APEX_WEB_SERVICE, [60-7](#)
 APEX_WORKFLOW, [61-1](#)
 APEX_ZIP, [62-1](#)
 DATE_POPUP function
 APEX_ITEM, [35-3, 35-4](#)
 DB_OPERATION_ALLOWED function
 APEX_PLUGIN_UTIL, [45-6](#)
 DB_SIGNATURE function
 APEX_INSTANCE_ADMIN, [32-19](#)
 DEBUG_DYNAMIC_ACTION procedure
 APEX_PLUGIN_UTIL, [45-7](#)
 DEBUG_PAGE_ITEM procedure signature 1
 APEX_PLUGIN_UTIL, [45-7](#)

DEBUG_PAGE_ITEM procedure signature 2
 APEX_PLUGIN_UTIL, [45-8](#)
 DEBUG_PROCESS procedure
 APEX_PLUGIN_UTIL, [45-9](#)
 DEBUG_REGION procedure signature 1
 APEX_PLUGIN_UTIL, [45-9](#)
 DEBUG_REGION procedure signature 2
 APEX_PLUGIN_UTIL, [45-10](#)
 DECODE function
 APEX_JWT, [38-3](#)
 DEFINE_USER_SESSION procedure
 APEX_CUSTOM_AUTH, [18-3](#)
 DEL_AD_COLUMN procedure
 APEX_UI_DEFAULT_UPDATE, [58-4](#)
 DEL_AD_SYNONYM procedure
 APEX_UI_DEFAULT_UPDATE, [58-4](#)
 DEL_COLUMN procedure
 APEX_UI_DEFAULT_UPDATE, [58-5](#)
 DEL_GROUP procedure
 APEX_UI_DEFAULT_UPDATE, [58-5](#)
 DEL_TABLE procedure
 APEX_UI_DEFAULT_UPDATE, [58-6](#)
 DELEGATE_TASK procedure
 APEX_APPROVAL, [9-12](#)
 APEX_HUMAN_TASK, [31-10](#)
 DELETE_ALL_COLLECTIONS procedure
 APEX_COLLECTION, [15-19](#)
 DELETE_ALL_COLLECTIONS_SESSION procedure
 APEX_COLLECTION, [15-19](#)
 DELETE_COLLECTION procedure
 APEX_COLLECTION, [15-20](#)
 DELETE_FEEDBACK procedure
 APEX_UTIL, [59-21](#)
 DELETE_FEEDBACK_ATTACHMENT procedure
 APEX_UTIL, [59-22](#)
 DELETE_GEOM_METADATA procedure
 APEX_SPATIAL, [54-3](#)
 DELETE_LANGUAGE_MAPPING procedure
 APEX_LANG, [39-4](#)
 DELETE_MEMBER procedure
 APEX_COLLECTION, [15-20](#)
 DELETE_MEMBERS procedure
 APEX_COLLECTION, [15-21](#)
 DELETE_MESSAGE procedure
 APEX_LANG, [39-5](#)
 DELETE_REPORT procedure
 APEX_IG, [33-7](#)
 APEX_IR, [34-9](#)
 DELETE_SESSION procedure
 APEX_SESSION, [51-4](#)
 DELETE_SUBSCRIPTION procedure
 APEX_IR, [34-10](#)
 DELETE_USER_GROUP procedure signature 1
 APEX_UTIL, [59-22](#)

DELETE_USER_GROUP procedure signature 2
 APEX_UTIL, 59-23

DESCRIBE_QUERY function signature 1
 APEX_EXEC, 27-35

DESCRIBE_QUERY function signature 2
 APEX_EXEC, 27-35

DETACH procedure
 APEX_SESSION, 51-3

DIFF function
 APEX_STRING_UTIL, 56-1

DISABLE procedure
 APEX_AUTOMATION, 12-2
 APEX_DEBUG, 23-2
 APEX_REST_SOURCE_SYNC, 49-1

DISABLE_DBMS_OUTPUT procedure
 APEX_DEBUG, 23-3

DISABLE_USER_STYLE procedure
 APEX_THEME, 57-2

DISCOVER function
 APEX_DATA_PARSER, 22-3

DISPLAY_AND_SAVE function
 APEX_ITEM, 35-6

DOES_EXIST function
 APEX_JSON, 37-5

DOWNLOAD procedure
 APEX_DATA_EXPORT, 20-11

DOWNLOAD procedure signature 1
 APEX_HTTP, 30-1

DOWNLOAD procedure signature 2
 APEX_HTTP, 30-2

DOWNLOAD_PRINT_DOCUMENT procedure
 signature 1
 APEX_UTIL, 59-24

DOWNLOAD_PRINT_DOCUMENT procedure
 signature 2
 APEX_UTIL, 59-24

DOWNLOAD_PRINT_DOCUMENT procedure
 signature 3
 APEX_UTIL, 59-25

DOWNLOAD_PRINT_DOCUMENT procedure
 signature 4
 APEX_UTIL, 59-27

DROP_CLOUD_CREDENTIAL procedure
 APEX_INSTANCE_ADMIN, 32-20

DROP_CREDENTIAL procedure
 APEX_CREDENTIAL, 16-4

DYNAMIC_SYNCHRONIZE_DATA procedure
 APEX_REST_SOURCE_SYNC, 49-2

E

EDIT_USER procedure
 APEX_UTIL, 59-28

email
 sending as an attachment, 41-2, 41-4

email (*continued*)
 sending messages in APEX_MAIL_QUEUE, 41-8
 sending outbound, 41-14

EMIT_LANGUAGE_SELECTOR_LIST procedure
 APEX_LANG, 39-6

ENABLE procedure
 APEX_AUTOMATION, 12-3
 APEX_DEBUG, 23-3
 APEX_REST_SOURCE_SYNC, 49-3

ENABLE_DBMS_OUTPUT procedure
 APEX_DEBUG, 23-5

ENABLE_DYNAMIC_GROUPS procedure
 APEX_AUTHORIZATION, 11-1

ENABLE_USER_STYLE procedure
 APEX_THEME, 57-3

ENCODE function
 APEX_JWT, 38-1

END_USER_ACCOUNT_DAYS_LEFT function
 APEX_UTIL, 59-32

ENQUOTE_LITERAL function
 APEX_EXEC, 27-37

ENQUOTE_NAME function
 APEX_EXEC, 27-38

ENTER procedure
 APEX_DEBUG, 23-4

error handling function
 APEX_ERROR, 25-3

ERROR procedure
 APEX_DEBUG, 23-6

ESCAPE function
 APEX_JAVASCRIPT, 36-11
 APEX_PLUGIN_UTIL, 45-11

EXECUTE_DML procedure
 APEX_EXEC, 27-38

EXECUTE for query context procedure
 APEX_AUTOMATION, 12-5

EXECUTE procedure signature 1
 APEX_AUTOMATION, 12-3

EXECUTE procedure signature 2
 APEX_AUTOMATION, 12-4

EXECUTE_PLSQL procedure signature 1
 APEX_EXEC, 27-39

EXECUTE_PLSQL procedure signature 2
 APEX_EXEC, 27-40

EXECUTE_PLSQL_CODE procedure
 APEX_PLUGIN_UTIL, 45-11

EXECUTE_REMOTE_PLSQL procedure
 signature 1
 APEX_EXEC, 27-41

EXECUTE_REMOTE_PLSQL procedure
 signature 2
 APEX_EXEC, 27-42

EXECUTE_REST_SOURCE procedure signature 1
 APEX_EXEC, 27-42

EXECUTE_REST_SOURCE procedure signature
2
APEX_EXEC, 27-44

EXECUTE_WEB_SOURCE procedure
APEX_EXEC, 27-44

EXIT procedure
APEX_AUTOMATION, 12-6

EXPIRE_END_USER_ACCOUNT procedure
APEX_UTIL, 59-33

EXPIRE_WORKSPACE_ACCOUNT procedure
APEX_UTIL, 59-33

export file
of workspace, 59-34

EXPORT function
APEX_DATA_EXPORT, 20-12

EXPORT_BLUEPRINT function
APEX_DG_DATA_GEN, 24-11

EXPORT_DATA function
APEX_REGION, 48-2

EXPORT_SAVED_REPORTS function
APEX_IR, 34-10

EXPORT_USERS procedure
APEX_UTIL, 59-34

EXTRACT_CONSTRAINT_NAME function
APEX_ERROR, 25-11

F

FEEDBACK_ENABLED function
APEX_UTIL, 59-35

FETCH_APP_ITEM function
APEX_UTIL, 59-35

FETCH_USER procedure signature 1
APEX_UTIL, 59-36

FETCH_USER procedure signature 2
APEX_UTIL, 59-39

FETCH_USER procedure signature 3
APEX_UTIL, 59-41

file repository
downloading files, 59-55
obtaining primary key, 59-56

FIND_EMAIL_ADDRESSES function
APEX_STRING_UTIL, 56-2

FIND_EMAIL_FROM function
APEX_STRING_UTIL, 56-3

FIND_EMAIL_SUBJECT function
APEX_STRING_UTIL, 56-4

FIND_IDENTIFIERS function
APEX_STRING_UTIL, 56-4

FIND_LINKS function
APEX_STRING_UTIL, 56-5

FIND_PATHS_LIKE function
APEX_JSON, 37-6

FIND_PHRASES function
APEX_STRING_UTIL, 56-6

FIND_SECURITY_GROUP_ID function
APEX_UTIL, 59-44

FIND_TAGS function
APEX_STRING_UTIL, 56-7

FIND_WORKSPACE function
APEX_UTIL, 59-44

FINISH procedure
APEX_ZIP, 62-3

flexible remote servers, 44-1

FLUSH procedure
APEX_JSON, 37-7

FORMAT function
APEX_STRING, 55-2

FREE_OUTPUT procedure
APEX_JSON, 37-8

FREE_WORKSPACE_APP_IDS procedure
APEX_INSTANCE_ADMIN, 32-20

G

G_Fnn arrays, 6-1

GENERATE function signature 1
APEX_AI, 3-3

GENERATE function signature 2
APEX_AI, 3-4

GENERATE_APPLICATION_ID procedure
APEX_APPLICATION_INSTALL, 8-7

GENERATE_DATA procedure
APEX_DG_DATA_GEN, 24-12, 24-13

GENERATE_DATA_INTO_COLLECTION
procedure
APEX_DG_DATA_GEN, 24-15

GENERATE_DOCUMENT function signature 1
APEX_PRINT, 46-2

GENERATE_DOCUMENT function signature 2
APEX_PRINT, 46-3

GENERATE_DOCUMENT function signature 3
APEX_PRINT, 46-4

GENERATE_DOCUMENT function signature 4
APEX_PRINT, 46-5

GENERATE_OFFSET procedure
APEX_APPLICATION_INSTALL, 8-7

GENERATE_PUSH_CREDENTIALS procedure
APEX_PWA, 47-1

GENERATE_REQUEST_BODY function
APEX_WEB_SERVICE, 60-12

GET function
APEX_EXEC, 27-46

GET_ACCOUNT_LOCKED_STATUS function
APEX_UTIL, 59-45

GET_AJAX_IDENTIFIER function
APEX_PLUGIN, 44-15

GET_ALL_USER_ATTRIBUTES procedure
APEX_LDAP, 40-2

GET_APEX_OWNER function
APEX_UTIL, 59-46

GET_APPLICATION function
 APEX_EXPORT, [28-1](#)

GET_APPLICATION_ALIAS function
 APEX_APPLICATION_ADMIN, [7-2](#)
 APEX_APPLICATION_INSTALL, [8-8](#)

GET_APPLICATION_ID function
 APEX_APPLICATION_INSTALL, [8-8](#)

GET_APPLICATION_NAME function
 APEX_APPLICATION_ADMIN, [7-3](#)
 APEX_APPLICATION_INSTALL, [8-9](#)

GET_APPLICATION_STATUS function
 APEX_APPLICATION_ADMIN, [7-4](#)
 APEX_UTIL, [59-46](#)

GET_APPLICATION_VERSION function
 APEX_APPLICATION_ADMIN, [7-4](#)

GET_ARRAY_ROW_DML_OPERATION function
 APEX_EXEC, [27-48](#)

GET_ARRAY_ROW_VERSION_CHECKSUM function
 APEX_EXEC, [27-49](#)

GET_ATTRIBUTE function
 APEX_UTIL, [59-47](#)

GET_ATTRIBUTE_AS_NUMBER function
 APEX_PLUGIN_UTIL, [45-12](#)

GET_AUTHENTICATION_RESULT function
 APEX_UTIL, [59-48](#)

GET_AUTHENTICATION_SCHEME function
 APEX_APPLICATION_ADMIN, [7-5](#)
 APEX_APPLICATION_INSTALL, [8-10](#)

GET_AUTO_INSTALL_SUP_OBJ function
 APEX_APPLICATION_INSTALL, [8-10](#)

GET_BLOB_FILE_SRC function
 APEX_UTIL, [59-48](#)

GET_BLUEPRINT_ID function
 APEX_DG_DATA_GEN, [24-16](#)

GET_BOOLEAN function
 APEX_JSON, [37-8](#)

GET_BP_TABLE_ID function
 APEX_DG_DATA_GEN, [24-17](#)

GET_BUILD_OPTION_STATUS function
 signature 1
 APEX_APPLICATION_ADMIN, [7-5](#)
 APEX_UTIL, [59-50](#)

GET_BUILD_OPTION_STATUS function
 signature 2
 APEX_APPLICATION_ADMIN, [7-6](#)
 APEX_UTIL, [59-51](#)

GET_BUILD_STATUS function
 APEX_APPLICATION_ADMIN, [7-7](#)
 APEX_APPLICATION_INSTALL, [8-11](#)

GET_CALLBACK_URL procedure
 APEX_AUTHENTICATION, [10-5](#)

GET_CLOB function
 APEX_JSON, [37-9](#)
 APEX_SESSION_STATE, [52-1](#)

GET_CLOB_OUTPUT function
 APEX_JSON, [37-10](#)

GET_CODE128_PNG function
 APEX_BARCODE, [14-1](#)

GET_CODE128_SVG function
 APEX_BARCODE, [14-2](#)

GET_COLUMN function
 APEX_EXEC, [27-50](#)

GET_COLUMN_COUNT function
 APEX_EXEC, [27-50](#)

GET_COLUMN_POSITION function
 APEX_EXEC, [27-51](#)

GET_COLUMNS function
 APEX_DATA_PARSER, [22-5](#)
 APEX_EXEC, [27-50](#)

GET_COOKIE_PROPS procedure
 APEX_CUSTOM_AUTH, [18-3](#)

GET_COUNT function
 APEX_JSON, [37-11](#)

GET_CSV_ENCLOSED_BY function
 APEX_ESCAPE, [26-4](#)

GET_CSV_SEPARATED_BY function
 APEX_ESCAPE, [26-5](#)

GET_CURRENT_DATABASE_TYPE function
 APEX_PLUGIN_UTIL, [45-13](#)

GET_CURRENT_EXECUTION function
 APEX_BACKGROUND_PROCESS, [13-4](#)

GET_CURRENT_USER_ID function
 APEX_UTIL, [59-51](#)

GET_DATA function signature 1
 APEX_PLUGIN_UTIL, [45-13](#)

GET_DATA function signature 2
 APEX_PLUGIN_UTIL, [45-15](#)

GET_DATA_TYPE function
 APEX_EXEC, [27-51](#)

GET_DATA2 function signature 1
 APEX_PLUGIN_UTIL, [45-17](#)

GET_DATA2 function signature 2
 APEX_PLUGIN_UTIL, [45-20](#)

GET_DATE function
 APEX_JSON, [37-12](#)

GET_DEFAULT_SCHEMA function
 APEX_UTIL, [59-52](#)

GET_DIR_ENTRIES function
 APEX_ZIP, [62-3](#)

GET_DISPLAY_DATA function signature 1
 APEX_PLUGIN_UTIL, [45-22](#)

GET_DISPLAY_DATA function signature 2
 APEX_PLUGIN_UTIL, [45-23](#)

GET_DML_STATUS_CODE function
 APEX_EXEC, [27-52](#)

GET_DML_STATUS_MESSAGE function
 APEX_EXEC, [27-53](#)

GET_DOMAIN function
 APEX_STRING_UTIL, [56-8](#)

GET_EAN8_PNG function
 APEX_BARCODE, [14-3](#)

GET_EAN8_SVG function
 APEX_BARCODE, [14-4](#)

GET_EDITION function
 APEX_UTIL, [59-53](#)

GET_ELEMENT_ATTRIBUTES function
 APEX_PLUGIN_UTIL, [45-25](#)

GET_EXAMPLE function
 APEX_DG_DATA_GEN, [24-18](#)

GET_EXECUTION function
 APEX_BACKGROUND_PROCESS, [13-5](#)

GET_FEEDBACK function
 APEX_EXPORT, [28-4](#)

GET_FEEDBACK_FOLLOW_UP function
 APEX_UTIL, [59-54](#)

GET_FILE procedure
 APEX_UTIL, [59-55](#)

GET_FILE_CONTENT function signature 1
 (deprecated)
 APEX_ZIP, [62-5](#)

GET_FILE_CONTENT function signature 2
 APEX_ZIP, [62-5](#)

GET_FILE_EXTENSION function
 APEX_STRING_UTIL, [56-8](#)

GET_FILE_ID function
 APEX_UTIL, [59-56](#)

GET_FILE_PROFILE function
 APEX_DATA_LOADING, [19-1](#)
 APEX_DATA_PARSER, [22-6](#)

GET_FILE_STORAGE function
 APEX_APPLICATION_ADMIN, [7-8](#)

GET_FILE_TYPE function
 APEX_DATA_PARSER, [22-8](#)

GET_FILES function
 APEX_ZIP, [62-6](#)

GET_FIRST_NAME function
 APEX_UTIL, [59-57](#)

GET_FIRST_ORA_ERROR_TEXT procedure
 APEX_ERROR, [25-11](#)

GET_GLOBAL_NOTIFICATION function
 APEX_APPLICATION_ADMIN, [7-7](#)
 APEX_UTIL, [59-57](#)

GET_GRANTOR_WORKSPACE function
 APEX_EXTENSION, [29-2](#)

GET_GROUP_ID function
 APEX_UTIL, [59-59](#)

GET_GROUP_NAME function
 APEX_UTIL, [59-60](#)

GET_GROUPS_USER_BELONGS_TO function
 APEX_UTIL, [59-58](#)

GET_HASH function
 APEX_UTIL, [59-60](#)

GET_HIGH_CONTRAST_MODE_TOGGLE function
 APEX_UTIL, [59-61](#)

GET_HTML_ATTR function
 APEX_PLUGIN_UTIL, [45-27](#)

GET_ID function signature 1
 APEX_REGION, [48-4](#)

GET_ID function signature 2
 APEX_REGION, [48-5](#)

GET_IMAGE_PREFIX function
 APEX_APPLICATION_ADMIN, [7-8](#)
 APEX_APPLICATION_INSTALL, [8-11](#)

GET_IMAGES_URL function
 APEX_MAIL, [41-5](#)

GET_INFO function
 APEX_APPLICATION_INSTALL, [8-12](#)

GET_INITIALIZES function
 APEX_STRING, [55-3](#)

GET_INPUT_NAME_FOR_PAGE_ITEM function
 APEX_PLUGIN, [44-16](#)

GET_INSTANCE_URL function
 APEX_MAIL, [41-6](#)

GET_KEEP_BACKGROUND_EXECS function
 APEX_APPLICATION_INSTALL, [8-14](#)

GET_KEEP_SESSIONS function
 APEX_APPLICATION_INSTALL, [8-14](#)

GET_LANGUAGE_SELECTOR_LIST function
 APEX_LANG, [39-7](#)

GET_LAST_MESSAGE_ID function
 APEX_DEBUG, [23-7](#)

GET_LAST_NAME function
 APEX_UTIL, [59-62](#)

GET_LAST_RUN function
 APEX_AUTOMATION, [12-6](#)

GET_LAST_RUN_TIMESTAMP function
 APEX_AUTOMATION, [12-7](#)

GET_LAST_SYNC_TIMESTAMP function
 APEX_REST_SOURCE_SYNC, [49-3](#)

GET_LAST_VIEWED_REPORT_ID function
 APEX_IG, [33-8](#)
 APEX_IR, [34-11](#)

GET_LDAP_PROPS procedure
 APEX_CUSTOM_AUTH, [18-4](#)

GET_LOGIN_USERNAME_COOKIE_LOGIN function
 APEX_AUTHENTICATION, [10-5](#)

GET_LOV_ACTIVITY_STATE function
 APEX_WORKFLOW, [61-5](#)

GET_LOV_PRIORITY function
 APEX_APPROVAL, [9-13](#)
 APEX_HUMAN_TASK, [31-11](#)

GET_LOV_STATE function
 APEX_APPROVAL, [9-13](#)
 APEX_HUMAN_TASK, [31-11](#)

GET_LOV_TYPE function
 APEX_HUMAN_TASK, [31-12](#)

GET_LOV_WORKFLOW_STATE function
 APEX_WORKFLOW, [61-6](#)

GET_MAX_SCHEDULER_JOBS function
 APEX_APPLICATION_ADMIN, 7-9
 APEX_APPLICATION_INSTALL, 8-15

GET_MEMBER_MD5 function
 APEX_COLLECTION, 15-22

GET_MEMBERS function
 APEX_JSON, 37-13

GET_MESSAGE function
 APEX_LANG, 39-7

GET_NEXT_PURGE_TIMESTAMP function
 APEX_APPROVAL, 9-14
 APEX_HUMAN_TASK, 31-12
 APEX_WORKFLOW, 61-6

GET_NEXT_SESSION_ID function
 APEX_CUSTOM_AUTH, 18-5

GET_NO_PROXY_DOMAINS function
 APEX_APPLICATION_ADMIN, 7-10
 APEX_APPLICATION_INSTALL, 8-15

GET_NUMBER function
 APEX_JSON, 37-14
 APEX_SESSION_STATE, 52-2

GET_NUMERIC_SESSION_STATE function
 APEX_UTIL, 59-63

GET_OFFSET function
 APEX_APPLICATION_INSTALL, 8-16

GET_ORDERBY_NULLS_SUPPORT function
 APEX_PLUGIN_UTIL, 45-27

GET_PAGE_MODE function
 APEX_PAGE, 43-1

GET_PAGE_VIEW_ID function
 APEX_DEBUG, 23-8

GET_PARAMETER function
 APEX_EXEC, 27-53
 APEX_INSTANCE_ADMIN, 32-21

GET_PARSING_SCHEMA function
 APEX_APPLICATION_ADMIN, 7-10

GET_PASS_ECID function
 APEX_APPLICATION_ADMIN, 7-11
 APEX_APPLICATION_INSTALL, 8-16

GET_PLSQL_EXPR_RESULT_BOOLEAN function
 APEX_PLUGIN_UTIL, 45-28

GET_PLSQL_EXPR_RESULT_CLOB function
 APEX_PLUGIN_UTIL, 45-29

GET_PLSQL_EXPRESSION_RESULT function
 APEX_PLUGIN_UTIL, 45-29

GET_PLSQL_FUNC_RESULT_BOOLEAN function
 APEX_PLUGIN_UTIL, 45-30

GET_PLSQL_FUNC_RESULT_CLOB function
 APEX_PLUGIN_UTIL, 45-31

GET_PLSQL_FUNCTION_RESULT function
 APEX_PLUGIN_UTIL, 45-32

GET_POSITION_IN_LIST function
 APEX_PLUGIN_UTIL, 45-32

GET_PREFERENCE function
 APEX_UTIL, 59-64

GET_PRINT_CONFIG procedure
 APEX_DATA_EXPORT, 20-14

GET_PRINT_DOCUMENT function signature 1
 APEX_UTIL, 59-65

GET_PRINT_DOCUMENT function signature 2
 APEX_UTIL, 59-65

GET_PRINT_DOCUMENT function signature 3
 APEX_UTIL, 59-66

GET_PRINT_DOCUMENT function signature 4
 APEX_UTIL, 59-67

GET_PROXY function
 APEX_APPLICATION_INSTALL, 8-17

GET_PROXY_SERVER function
 APEX_APPLICATION_ADMIN, 7-11

GET_QRCODE_PNG function
 APEX_BARCODE, 14-5

GET_QRCODE_SVG function
 APEX_BARCODE, 14-6

GET_REMOTE_SERVER_AI_ATTRS function
 APEX_APPLICATION_INSTALL, 8-17

GET_REMOTE_SERVER_AI_HEADERS function
 APEX_APPLICATION_INSTALL, 8-18

GET_REMOTE_SERVER_AI_MODEL function
 APEX_APPLICATION_INSTALL, 8-19

GET_REMOTE_SERVER_BASE_URL function
 APEX_APPLICATION_INSTALL, 8-19

GET_REMOTE_SERVER_DEFAULT_DB function
 APEX_APPLICATION_INSTALL, 8-20

GET_REMOTE_SERVER_HTTPS_HOST function
 APEX_APPLICATION_INSTALL, 8-21

GET_REMOTE_SERVER_SQL_MODE function
 APEX_APPLICATION_INSTALL, 8-21

GET_REPORT function
 APEX_IR, 34-12

GET_REQUEST_HEADER function
 APEX_WEB_SERVICE, 60-12

GET_REST_SOURCE_CATALOG_GROUP function
 APEX_APPLICATION_INSTALL, 8-22

GET_ROW_VERSION_CHECKSUM function
 APEX_EXEC, 27-54

GET_SCHEDULER_JOB_NAME function
 APEX_AUTOMATION, 12-8

GET_SCHEMA function
 APEX_APPLICATION_INSTALL, 8-23

GET_SCHEMAS function
 APEX_INSTANCE_ADMIN, 32-21

GET_SCREEN_READER_MODE_TOGGLE function
 APEX_UTIL, 59-68

GET_SDO_GEOMETRY function
 APEX_JSON, 37-15

GET_SEARCH_STRING function
APEX_PLUGIN_UTIL, 45-33

GET_SEARCHABLE_PHRASES function
APEX_STRING, 55-4

GET_SECURITY_GROUP_ID function
APEX_CUSTOM_AUTH, 18-6

GET_SESSION_ID function
APEX_CUSTOM_AUTH, 18-6

GET_SESSION_ID_FROM_COOKIE
APEX_CUSTOM_AUTH, 18-6

GET_SESSION_LANG function
APEX_UTIL, 59-68

GET_SESSION_STATE function
APEX_UTIL, 59-69

GET_SESSION_TERRITORY function
APEX_UTIL, 59-70

GET_SESSION_TIME_ZONE function
APEX_UTIL, 59-70

GET_SINCE function signature 1
APEX_UTIL, 59-71

GET_SINCE function signature 2
APEX_UTIL, 59-72

GET_SLUG function
APEX_STRING_UTIL, 56-9

GET_SUPPORTING_OBJECT_SCRIPT function
APEX_UTIL, 59-72

GET_SUPPORTING_OBJECT_SCRIPT
procedure
APEX_UTIL, 59-73

GET_SYNC_TABLE_DEFINITION_SQL function
APEX_REST_SOURCE_SYNC, 49-4

GET_T_NUMBER function
APEX_JSON, 37-16

GET_T_VARCHAR2 function
APEX_JSON, 37-17

GET_TASK_DELEGATES function
APEX_APPROVAL, 9-15
APEX_HUMAN_TASK, 31-13

GET_TASK_HISTORY function
APEX_APPROVAL, 9-15
APEX_HUMAN_TASK, 31-13

GET_TASK_PARAMETER_OLD_VALUE function
APEX_APPROVAL, 9-17
APEX_HUMAN_TASK, 31-14

GET_TASK_PARAMETER_VALUE function
APEX_APPROVAL, 9-18
APEX_HUMAN_TASK, 31-14

GET_TASK_PRIORITIES function
APEX_APPROVAL, 9-19
APEX_HUMAN_TASK, 31-15

GET_TASKS function
APEX_APPROVAL, 9-19
APEX_HUMAN_TASK, 31-16

GET_THEME_ID function
APEX_APPLICATION_INSTALL, 8-23

GET_TIMESTAMP function
APEX_SESSION_STATE, 52-2

GET_TOTAL_ROW_COUNT function
APEX_EXEC, 27-54

GET_UI_TYPE function (deprecated)
APEX_PAGE, 43-1

GET_URL function
APEX_PAGE, 43-2

GET_USER function
APEX_CUSTOM_AUTH, 18-7

GET_USER_ATTRIBUTES procedure
APEX_LDAP, 40-3

GET_USER_ID function
APEX_UTIL, 59-74

GET_USER_ROLES function
APEX_UTIL, 59-75

GET_USER_STYLE Function
APEX_THEME, 57-4

GET_USERNAME
APEX_CUSTOM_AUTH, 18-7

GET_USERNAME function
APEX_UTIL, 59-76

GET_VALUE function
APEX_APP_SETTING, 5-1
APEX_JSON, 37-19
APEX_SESSION_STATE, 52-2

GET_VALUE_AS_VARCHAR2 function
APEX_PLUGIN_UTIL, 45-34

GET_VALUE_KIND function
APEX_JSON, 37-19

GET_VARCHAR2 function
APEX_JSON, 37-21
APEX_SESSION_STATE, 52-3

GET_VARIABLE_CLOB_VALUE function
APEX_WORKFLOW, 61-7

GET_VARIABLE_VALUE function
APEX_WORKFLOW, 61-7

GET_VECTOR_EMBEDDINGS function signature
1
APEX_AI, 3-5

GET_VECTOR_EMBEDDINGS function signature
2
APEX_AI, 3-6

GET_VECTOR_EMBEDDINGS function signature
3
APEX_AI, 3-6

GET_WEB_SOURCE_OPERATION function
APEX_PLUGIN_UTIL, 45-35

GET_WEIGHTED_INLINE_DATA function
APEX_DG_DATA_GEN, 24-18

GET_WORKFLOW_STATE function
APEX_WORKFLOW, 61-8

GET_WORKFLOWS function
APEX_WORKFLOW, 61-9

GET_WORKSPACE function
APEX_EXPORT, 28-5

GET_WORKSPACE_FILES function
APEX_EXPORT, [28-4](#)

GET_WORKSPACE_ID function
APEX_APPLICATION_INSTALL, [8-24](#)

GET_WORKSPACE_PARAMETER procedure
APEX_INSTANCE_ADMIN, [32-22](#)

GET_XLIFF_DOCUMENT function
APEX_LANG, [39-8](#)

GET_XLSX_WORKSHEETS function
APEX_DATA_PARSER, [22-9](#)

global constants
APEX_AI, [3-1](#)
APEX_APP_OBJECT_DEPENDENCY, [4-1](#)
APEX_APPLICATION, [6-3](#)
APEX_APPLICATION_ADMIN, [7-2](#)
APEX_APPROVAL, [9-2](#)
APEX_AUTHENTICATION, [10-1](#)
APEX_BACKGROUND_PROCESS, [13-1](#)
APEX_DATA_EXPORT, [20-1](#)
APEX_DATA_PARSER, [22-1](#)
APEX_DEBUG, [23-2](#)
APEX_ESCAPE, [26-2](#)
APEX_EXEC, [27-6](#)
APEX_HUMAN_TASK, [31-2](#)
APEX_IR, [34-1](#)
APEX_JSON, [37-3](#)
APEX_MARKDOWN, [42-1](#)
APEX_PAGE, [43-1](#)
APEX_PLUGIN, [44-2](#)
APEX_PRINT, [46-1](#)
APEX_SESSION_STATE, [52-1](#)
APEX_SHARED_COMPONENT, [53-1](#)
APEX_WEB_SERVICE, [60-7](#)
APEX_WORKFLOW, [61-1](#)

global variables
APEX_APPLICATION, [6-3](#)
APEX_WEB_SERVICE, [60-7](#)

GRANT_EXTENSION_WORKSPACE procedure
APEX_INSTANCE_ADMIN, [32-23](#)

GREP function signature 1
APEX_STRING, [55-5](#)

GREP function signature 2
APEX_STRING, [55-6](#)

GREP function signature 3
APEX_STRING, [55-7](#)

H

HANDLE_TASK_DEADLINES procedure
APEX_APPROVAL, [9-21](#)
APEX_HUMAN_TASK, [31-18](#)

HAS_ERROR function
APEX_EXEC, [27-55](#)

HAS_MORE_ARRAY_ROWS function
APEX_EXEC, [27-55](#)

HAS_MORE_ROWS function
APEX_EXEC, [27-56](#)

HAS_PUSH_SUBSCRIPTION function
APEX_PWA, [47-1](#)

HAS_TASK_PARAM_CHANGED function
APEX_APPROVAL, [9-22](#)
APEX_HUMAN_TASK, [31-18](#)

HAS_USER_ANY_ROLES function
APEX_ACL, [2-2](#)

HAS_USER_ROLE function
APEX_ACL, [2-3](#)

HAVE_ERRORS_OCCURRED function
APEX_ERROR, [25-12](#)

HELP Procedure
APEX_APPLICATION, [6-3](#)

HIDDEN function
APEX_ITEM, [35-6](#)

HOST_URL function
APEX_UTIL, [59-77](#)

HTML function
APEX_ERROR, [26-5](#)

HTML_ALLOWLIST function
APEX_ESCAPE, [26-7](#)

HTML_ALLOWLIST_CLOB function
APEX_ESCAPE, [26-7](#)

HTML_ATTRIBUTE function
APEX_ESCAPE, [26-8](#)

HTML_ATTRIBUTE_CLOB function
APEX_ESCAPE, [26-10](#)

HTML_CLOB function
APEX_ESCAPE, [26-10](#)

HTML_TRUNC function signature 1
APEX_ESCAPE, [26-12](#)

HTML_TRUNC function signature 2
APEX_ESCAPE, [26-13](#)

I

import application, [8-4](#)

Import Data Types
APEX_APPLICATION_INSTALL, [8-3](#)

import script, [8-4](#)

IMPORT_BLUEPRINT procedure
APEX_DG_DATA_GEN, [24-19](#)

IMPORT_SAVED_REPORTS procedure
APEX_IR, [34-13](#)

INDEX_OF function signature 1
APEX_STRING, [55-8](#)

INDEX_OF function signature 2
APEX_STRING, [55-8](#)

INFO procedure
APEX_DEBUG, [23-8](#)

INITIALIZE_OUTPUT procedure
APEX_JSON, [37-22, 37-23](#)

INSERT_GEO_METADATA procedure
APEX_SPATIAL, [54-4](#)

INSERT_GEOM_METADATA_LONLAT procedure
APEX_SPATIAL, 54-5

INSTALL procedure
APEX_APPLICATION_INSTALL, 8-24

installation, 8-1

INT_ERROR_RESULT function
APEX_ERROR, 25-12

IR_CLEAR procedure
APEX_UTIL, 59-79

IR_DELETE_REPORT procedure
APEX_UTIL, 59-80

IR_DELETE_SUBSCRIPTION procedure
APEX_UTIL, 59-81

IR_FILTER procedure
APEX_UTIL, 59-81

IR_RESET procedure
APEX_UTIL, 59-83

IS_ADMIN function
APEX_WORKFLOW, 61-10

IS_ALLOWED function
APEX_APPROVAL, 9-23
APEX_HUMAN_TASK, 31-19
APEX_WORKFLOW, 61-11

IS_AUTHENTICATED function
APEX_AUTHENTICATION, 10-6

IS_AUTHORIZED function
APEX_AUTHORIZATION, 11-2

IS_BUSINESS_ADMIN function
APEX_APPROVAL, 9-24
APEX_HUMAN_TASK, 31-20

IS_COMPONENT_USED function
APEX_PLUGIN_UTIL, 45-37

IS_DB_SIGNATURE_VALID function
APEX_INSTANCE_ADMIN, 32-24

IS_DESKTOP_UI function (deprecated)
APEX_PAGE, 43-3

IS_ENABLED function
APEX_AI, 3-7

IS_EQUAL function
APEX_PLUGIN_UTIL, 45-36

IS_GROUP_END function
APEX_EXEC, 27-57

IS_HIGH_CONTRAST_SESSION_YN function
APEX_UTIL, 59-84

IS_LOGIN_PASSWORD_VALID function
APEX_UTIL, 59-85

IS_MEMBER function
APEX_LDAP, 40-4

IS_OF_PARTICIPANT_TYPE function
APEX_APPROVAL, 9-25
APEX_HUMAN_TASK, 31-20
APEX_WORKFLOW, 61-12

IS_PUBLIC_USER function
APEX_AUTHENTICATION, 10-7

IS_READ_ONLY function
APEX_PAGE, 43-3

IS_READ_ONLY function (*continued*)
APEX_REGION, 48-6

IS_REMOTE_SQL_AUTH_VALID function
APEX_EXEC, 27-59

IS_ROLE_REMOVED_FROM_USER function
APEX_ACL, 2-4

IS_RUNNING function
APEX_AUTOMATION, 12-8
APEX_REST_SOURCE_SYNC, 49-5

IS_SCREEN_READER_SESSION function
APEX_UTIL, 59-86

IS_SCREEN_READER_SESSION_YN function
APEX_UTIL, 59-86

IS_SESSION_VALID function
APEX_CUSTOM_AUTH, 18-8

IS_USER_CONSENT_NEEDED function
APEX_AI, 3-7

J

JOIN function signature 1, 55-10

JOIN function signature 2
APEX_STRING, 55-11

JOIN_CLOB
APEX_STRING, 55-9

JOIN_CLOBS function
APEX_STRING, 55-10

JS_LITERAL function
APEX_ESCAPE, 26-14

JS_LITERAL_CLOB function
APEX_ESCAPE, 26-15

JSON function
APEX_ESCAPE, 26-16

JSON_CLOB function
APEX_ESCAPE, 26-16

JSON_TO_PROFILE function
APEX_DATA_PARSER, 22-10

K

KEYVAL_NUM function
APEX_UTIL, 59-87

KEYVAL_VC2 function
APEX_UTIL, 59-88

L

LANG function
APEX_LANG, 39-9

LDAP attributes
obtaining, 18-4

LDAP_DN function
APEX_ESCAPE, 26-17

LDAP_DNPREP function
APEX_CUSTOM_AUTH, 18-8

LDAP_SEARCH_FILTER function
APEX_ESCAPE, [26-18](#)
LOAD_DATA function
APEX_DATA_LOADING, [19-2](#), [19-3](#)
LOAD_SUPPORTING_OBJECT_DATA procedure
APEX_DATA_INSTALL, [21-1](#)
LOCK_ACCOUNT procedure
APEX_UTIL, [59-89](#), [59-139](#)
LOG_DBMS_OUTPUT procedure
APEX_DEBUG, [23-9](#)
LOG_ERROR procedure
APEX_AUTOMATION, [12-9](#)
LOG_INFO procedure
APEX_AUTOMATION, [12-10](#)
LOG_LONG_MESSAGE procedure
APEX_DEBUG, [23-10](#)
LOG_MESSAGE procedure
APEX_DEBUG, [23-11](#)
LOG_PAGE_SESSION_STATE procedure
APEX_DEBUG, [23-13](#)
LOG_WARN procedure
APEX_AUTOMATION, [12-10](#)
LOGIN procedure
APEX_AUTHENTICATION, [10-7](#)
APEX_CUSTOM_AUTH, [18-9](#)
LOGOUT procedure
APEX_CUSTOM_AUTH, [18-10](#)
LOGOUT Procedure
APEX_AUTHENTICATION, [10-8](#)

M

MAKE_REQUEST function signature 1
APEX_WEB_SERVICE, [60-13](#)
MAKE_REQUEST function signature 2
APEX_WEB_SERVICE, [60-15](#)
MAKE_REQUEST procedure signature 1
APEX_WEB_SERVICE, [60-16](#)
MAKE_REQUEST procedure signature 2
APEX_WEB_SERVICE, [60-18](#)
MAKE_REST_REQUEST function
APEX_WEB_SERVICE, [60-19](#), [60-20](#)
MAKE_REST_REQUEST procedure
APEX_PLUGIN_UTIL, [45-38](#), [45-39](#)
MD5_CHECKSUM function
APEX_ITEM, [35-7](#)
MD5_HIDDEN function
APEX_ITEM, [35-8](#)
MEMBER_OF function
APEX_LDAP, [40-6](#)
MEMBER_OF2 function
APEX_LDAP, [40-7](#)
MERGE_MEMBERS procedure
APEX_COLLECTION, [15-23](#)
MESSAGE function (deprecated)
APEX_LANG, [39-10](#)

MESSAGE procedure
APEX_DEBUG, [23-13](#)
MOVE_MEMBER_DOWN procedure
APEX_COLLECTION, [15-25](#)
MOVE_MEMBER_UP procedure
APEX_COLLECTION, [15-26](#)

N

NEXT_ARRAY_ROW function
APEX_EXEC, [27-59](#)
NEXT_CHUNK function
APEX_STRING, [55-11](#)
NEXT_ROW function
APEX_EXEC, [27-60](#)
NOOP function signature 1
APEX_ESCAPE, [26-19](#)
NOOP function signature 2
APEX_ESCAPE, [26-19](#)
NUMBER
APEX_MAIL, [41-9](#)

O

OAUTH_AUTHENTICATE procedure signature 1
APEX_WEB_SERVICE, [60-23](#)
OAUTH_AUTHENTICATE procedure signature 2
APEX_WEB_SERVICE, [60-24](#)
OAUTH_AUTHENTICATE_CREDENTIAL
procedure
APEX_WEB_SERVICE, [60-22](#)
OAUTH_GET_LAST_TOKEN function
APEX_WEB_SERVICE, [60-25](#)
OAUTH_SET_TOKEN procedure
APEX_WEB_SERVICE, [60-25](#)
OPEN_ARRAY procedure
APEX_JSON, [37-23](#)
OPEN_ARRAY procedure signature 1
APEX_EXEC, [27-60](#)
OPEN_ARRAY procedure signature 2
APEX_EXEC, [27-62](#)
OPEN_DUALITY_VIEW_DML_CONTEXT
function
APEX_EXEC, [27-63](#)
OPEN_JSON_SOURCE_DML_CONTEXT
function
APEX_EXEC, [27-65](#)
OPEN_LOCAL_DML_CONTEXT function
APEX_EXEC, [27-67](#)
OPEN_OBJECT procedure
APEX_JSON, [37-24](#)
OPEN_QUERY_CONTEXT function
APEX_REGION, [48-6](#)
OPEN_QUERY_CONTEXT function signature 1
APEX_EXEC, [27-70](#)

OPEN_QUERY_CONTEXT function signature 2
 APEX_EXEC, [27-74](#)
 OPEN_REMOTE_DML_CONTEXT function
 APEX_EXEC, [27-75](#)
 OPEN_REMOTE_SQL_QUERY function
 APEX_EXEC, [27-78](#)
 OPEN_REST_SOURCE_DML_CONTEXT function
 APEX_EXEC, [27-80](#)
 OPEN_REST_SOURCE_QUERY function
 APEX_EXEC, [27-83](#)
 OPEN_WEB_SOURCE_DML_CONTEXT function
 APEX_EXEC, [27-85](#)
 OPEN_WEB_SOURCE_QUERY function
 APEX_EXEC, [27-87](#)
 Oracle TEXT
 ADD_FILTER procedure, [27-21](#)

P

PAGE_ITEM_NAMES_TO_JQUERY function
 APEX_PLUGIN_UTIL, [45-41](#)
 parameter values
 APEX_INSTANCE_ADMIN, [32-2](#)
 PARSE function
 APEX_DATA_PARSER, [22-10](#)
 PARSE procedure signature 1
 APEX_JSON, [37-25](#)
 PARSE procedure signature 2
 APEX_JSON, [37-26](#)
 PARSE_REFETCH_RESPONSE function
 APEX_PLUGIN_UTIL, [45-42](#)
 PARSE_RESPONSE function
 APEX_WEB_SERVICE, [60-26](#)
 PARSE_RESPONSE_CLOB function
 APEX_WEB_SERVICE, [60-26](#)
 PARSE_XML function
 APEX_WEB_SERVICE, [60-27](#)
 PARSE_XML_CLOB function
 APEX_WEB_SERVICE, [60-28](#)
 password
 changing, [59-9](#)
 resetting and emailing, [59-101](#)
 PASSWORD_FIRST_USE_OCCURRED function
 APEX_UTIL, [59-89](#)
 PERSISTENT_AUTH_ENABLED function
 APEX_AUTHENTICATION, [10-9](#)
 PERSISTENT_COOKIES_ENABLED function
 APEX_AUTHENTICATION, [10-9](#)
 PHRASE_EXISTS function
 APEX_STRING_UTIL, [56-9](#)
 PLIST_DELETE procedure
 APEX_STRING, [55-12](#)
 PLIST_EXISTS function
 APEX_STRING, [55-13](#)

PLIST_GET function
 APEX_STRING, [55-14](#)
 PLIST_GET_KEY function
 APEX_STRING, [55-14](#)
 PLIST_PUSH procedure
 APEX_STRING, [55-15](#)
 PLIST_PUT function
 APEX_STRING, [55-16](#)
 PLIST_TO_JSON_CLOB function
 APEX_STRING, [55-16](#)
 plugins
 remote servers, [44-1](#)
 POINT function
 APEX_SPATIAL, [54-6](#)
 POPUP_FROM_LOV function
 APEX_ITEM, [35-9](#)
 POPUP_FROM_QUERY function
 APEX_ITEM, [35-11](#)
 POPUPKEY_FROM_LOV function
 APEX_ITEM, [35-12](#)
 POPUPKEY_FROM_QUERY function
 APEX_ITEM, [35-14](#)
 POST_LOGIN procedure
 APEX_AUTHENTICATION, [10-10](#)
 APEX_CUSTOM_AUTH, [18-10](#)
 PREPARE_TEMPLATE procedure
 APEX_MAIL, [41-7](#)
 PREPARE_URL function
 APEX_UTIL, [59-90](#)
 PREVIEW_BLUEPRINT procedure
 APEX_DG_DATA_GEN, [24-20](#)
 PRINT_DISPLAY_ONLY procedure signature 1
 APEX_PLUGIN_UTIL, [45-44](#)
 PRINT_DISPLAY_ONLY procedure signature 2
 APEX_PLUGIN_UTIL, [45-45](#)
 PRINT_ESCAPED_VALUE procedure signature 1
 APEX_PLUGIN_UTIL, [45-46](#)
 PRINT_ESCAPED_VALUE procedure signature 2
 APEX_PLUGIN_UTIL, [45-46](#)
 PRINT_HIDDEN procedure
 APEX_PLUGIN_UTIL, [45-47](#)
 PRINT_HIDDEN_IF_READONLY procedure
 APEX_PLUGIN_UTIL, [45-47](#)
 PRINT_JSON_HTTP_HEADER procedure
 APEX_PLUGIN_UTIL, [45-48](#)
 PRINT_LOV_AS_JSON procedure
 APEX_PLUGIN_UTIL, [45-49](#)
 PRINT_OPTION procedure
 APEX_PLUGIN_UTIL, [45-50](#)
 PRINT_READ_ONLY procedure signature 1
 APEX_PLUGIN_UTIL, [45-51](#)
 PRINT_READ_ONLY procedure signature 2
 APEX_PLUGIN_UTIL, [45-52](#)
 PRN procedure
 APEX_UTIL, [59-92](#)

PROCESS_DML_RESPONSE procedure
APEX_PLUGIN_UTIL, 45-53

PUBLIC_CHECK_AUTHORIZATION function
APEX_UTIL, 59-93

PUBLISH procedure
APEX_SHARED_COMPONENT, 53-2

PUBLISH_APPLICATION procedure
APEX_LANG, 39-11

PURGE_CACHE procedure
APEX_PAGE, 43-4
APEX_REGION, 48-8

PURGE_DUALITY_VIEW_CACHE procedure
APEX_EXEC, 27-89

PURGE_JSON_SOURCE_CACHE procedure
APEX_EXEC, 27-90

PURGE_REGIONS_BY_APP procedure
APEX_UTIL, 59-94

PURGE_REGIONS_BY_NAME procedure
APEX_UTIL, 59-94

PURGE_REGIONS_BY_PAGE procedure
APEX_UTIL, 59-95

PURGE_REST_SOURCE_CACHE procedure
APEX_EXEC, 27-90

PURGE_WEB_SOURCE_CACHE procedure
APEX_EXEC, 27-91

PUSH procedure signature 1
APEX_STRING, 55-17

PUSH procedure signature 2
APEX_STRING, 55-18

PUSH procedure signature 3
APEX_STRING, 55-18

PUSH procedure signature 4
APEX_STRING, 55-19

PUSH procedure signature 5
APEX_STRING, 55-19

PUSH procedure signature 6
APEX_STRING, 55-20

PUSH procedure signature 7
APEX_STRING, 55-21

PUSH_QUEUE procedure
APEX_MAIL, 41-8
APEX_PWA, 47-2

Q

QUERY_EXPERT_SEARCH function
APEX_SEARCH, 50-1

QUERY_SEARCH_ENGINE function
APEX_SEARCH, 50-2

R

RADIOGROUP function
APEX_ITEM, 35-15

RECTANGLE function
APEX_SPATIAL, 54-6

REDIRECT_URL procedure
APEX_UTIL, 59-96

REFRESH procedure
APEX_SHARED_COMPONENT, 53-2

REFRESH_PARTICIPANTS procedure
APEX_WORKFLOW, 61-12

REGEXP function
APEX_ESCAPE, 26-20

REJECT_TASK procedure
APEX_APPROVAL, 9-26
APEX_HUMAN_TASK, 31-21

RELEASE_TASK procedure
APEX_APPROVAL, 9-27
APEX_HUMAN_TASK, 31-22

remote servers, 44-1

REMOVE_ALL_USER_ROLES procedure
APEX_ACL, 2-8

REMOVE_APPLICATION procedure
APEX_APPLICATION_INSTALL, 8-26
APEX_INSTANCE_ADMIN, 32-25

REMOVE_AUTO_PROV_RESTRICTIONS procedure
APEX_INSTANCE_ADMIN, 32-25

REMOVE_BLUEPRINT procedure
APEX_DG_DATA_GEN, 24-20

REMOVE_COLUMN procedure
APEX_DG_DATA_GEN, 24-21

REMOVE_CURRENT_PERSISTENT_AUTH procedure
APEX_AUTHENTICATION, 10-10

REMOVE_DATA_SOURCE procedure
APEX_DG_DATA_GEN, 24-21

REMOVE_DEBUG_BY_APP procedure
APEX_DEBUG, 23-15

REMOVE_DEBUG_BY_VIEW procedure
APEX_DEBUG, 23-16

REMOVE_DEVELOPMENT_INSTANCES procedure
APEX_WORKFLOW, 61-13

REMOVE_MENU_ENTRY procedure
APEX_EXTENSION, 29-3

REMOVE_PERSISTENT_AUTH procedure
APEX_AUTHENTICATION, 10-11

REMOVE_POTENTIAL_OWNER procedure
APEX_APPROVAL, 9-27
APEX_HUMAN_TASK, 31-23

REMOVE_REQUEST_HEADER procedure
APEX_WEB_SERVICE, 60-29

REMOVE_SAVED_REPORT procedure
APEX_INSTANCE_ADMIN, 32-26

REMOVE_SAVED_REPORTS procedure
APEX_INSTANCE_ADMIN, 32-26

REMOVE_SCHEMA procedure
APEX_INSTANCE_ADMIN, 32-27

REMOVE_SCHEMA_EXCEPTION procedure
 APEX_INSTANCE_ADMIN, [32-27](#)

REMOVE_SCHEMA_EXCEPTIONS procedure
 APEX_INSTANCE_ADMIN, [32-28](#)

REMOVE_SESSION_MESSAGES procedure
 APEX_DEBUG, [23-16](#)

REMOVE_SORT_PREFERENCES procedure
 APEX_UTIL, [59-97](#)

REMOVE_SUBSCRIPTION procedure
 APEX_INSTANCE_ADMIN, [32-29](#)

REMOVE_TABLE procedure
 APEX_DG_DATA_GEN, [24-22](#)

REMOVE_TEMPLATE procedure
 APEX_PRINT, [46-6](#)

REMOVE_USER procedure
 APEX_UTIL, [59-97, 59-98](#)

REMOVE_USER_ROLE procedure signature 1
 APEX_ACL, [2-5](#)

REMOVE_USER_ROLE procedure signature 2
 APEX_ACL, [2-6](#)

REMOVE_WEB_ENTRY_POINT procedure
 APEX_INSTANCE_ADMIN, [32-30](#)

REMOVE_WORKSPACE procedure
 APEX_INSTANCE_ADMIN, [32-30](#)

REMOVE_WORKSPACE_EXCEPTIONS procedure
 APEX_INSTANCE_ADMIN, [32-31](#)

RENEW_TASK function
 APEX_APPROVAL, [9-28](#)
 APEX_HUMAN_TASK, [31-23](#)

REPLACE_SUBSTITUTIONS function
 APEX_PLUGIN_UTIL, [45-54](#)

REPLACE_USER_ROLES procedure signature 1
 APEX_ACL, [2-6](#)

REPLACE_USER_ROLES procedure signature 2
 APEX_ACL, [2-7](#)

REPLACE_WHITESPACE function
 APEX_STRING_UTIL, [56-10](#)

REPLY_TO_FEEDBACK procedure
 APEX_UTIL, [59-99](#)

REQUEST_MORE_INFORMATION procedure
 APEX_APPROVAL, [9-29](#)
 APEX_HUMAN_TASK, [31-24](#)

RESCHEDULE procedure
 APEX_AUTOMATION, [12-11](#)
 APEX_REST_SOURCE_SYNC, [49-5](#)

RESEQUENCE_BLUEPRINT procedure
 APEX_DG_DATA_GEN, [24-22](#)

RESEQUENCE_COLLECTION procedure
 APEX_COLLECTION, [15-27](#)

RESERVE_WORKSPACE_APP_IDS procedure
 APEX_INSTANCE_ADMIN, [32-32](#)

RESET procedure
 APEX_REGION, [48-8](#)

RESET_AUTHORIZATIONS procedure
 APEX_UTIL, [59-100](#)

RESET_CACHE procedure
 APEX_AUTHORIZATION, [11-3](#)

RESET_COLLECTION_CHANGED procedure
 APEX_COLLECTION, [15-27](#)

RESET_COLLECTION_CHANGED_ALL procedure
 APEX_COLLECTION, [15-28](#)

RESET_PASSWORD procedure
 APEX_UTIL, [59-100](#)

RESET_PW procedure
 APEX_UTIL, [59-101](#)

RESET_REPORT procedure signature 1
 APEX_IG, [33-9](#)
 APEX_IR, [34-15](#)

RESET_REPORT procedure signature 2
 APEX_IG, [33-9](#)
 APEX_IR, [34-15](#)

REST Data Source
 constants, [44-2](#)
 format, [44-2](#)

RESTful Web Service, [60-4](#)

RESTRICT_SCHEMA procedure
 APEX_INSTANCE_ADMIN, [32-33](#)

RESUME procedure
 APEX_WORKFLOW, [61-13](#)

Retrieving Cookies and HTTP Headers
 APEX_WEB_SERVICE, [60-5](#)

RETRY procedure
 APEX_WORKFLOW, [61-14](#)

REVOKE_EXTENSION_WORKSPACE procedure
 APEX_INSTANCE_ADMIN, [32-34](#)

REVOKE_USER_CONSENT procedure
 APEX_AI, [3-8](#)

REVOKE_USER_CONSENT_FOR_ALL procedure
 APEX_AI, [3-8](#)

S

SAML_CALLBACK procedure
 APEX_AUTHENTICATION, [10-12](#)

SAML_METADATA procedure
 APEX_AUTHENTICATION, [10-12](#)

SAVEKEY_NUM function
 APEX_UTIL, [59-102](#)

SAVEKEY_VC2 function
 APEX_UTIL, [59-103](#)

SCAN procedure
 APEX_APP_OBJECT_DEPENDENCY, [4-2](#)

SEARCH function
 APEX_LDAP, [40-8](#)
 APEX_SEARCH, [50-3](#)

SEED_TRANSLATIONS procedure
 APEX_LANG, [39-12](#)

SELECT_LIST function
 APEX_ITEM, [35-16](#)

SELECT_LIST_FROM_LOV function
 APEX_ITEM, [35-18](#)

SELECT_LIST_FROM_LOV_XL function
 APEX_ITEM, [35-19](#)

SELECT_LIST_FROM_QUERY function
 APEX_ITEM, [35-20](#)

SELECT_LIST_FROM_QUERY_XL function
 APEX_ITEM, [35-21](#)

SEND function
 APEX_MAIL, [41-12](#)

SEND function signature 1
 APEX_MAIL, [41-9](#)

SEND procedure signature 1
 APEX_MAIL, [41-14](#)

SEND procedure signature 2
 APEX_MAIL, [41-17](#)

SEND_LOGIN_USERNAME_COOKIE procedure
 APEX_AUTHENTICATION, [10-13](#)

SEND_PUSH_NOTIFICATION procedure
 APEX_PWA, [47-2](#)

sending email in queue
 APEX_MAIL_QUEUE, [41-8](#)

session cookies, [18-3](#)

session state
 fetching for current application, [59-35](#)
 removing for current page, [59-13](#)
 removing for current session, [59-13](#)
 setting, [59-121](#), [59-123](#)

SESSION_ID_EXISTS function
 APEX_CUSTOM_AUTH, [18-11](#)

SET_ALLOWED_URLS procedure
 APEX_CREDENTIAL, [16-5](#)

SET_APP_BUILD_STATUS Procedure
 APEX_UTIL, [59-104](#)

SET_APPLICATION_ALIAS procedure
 APEX_APPLICATION_ADMIN, [7-12](#)
 APEX_APPLICATION_INSTALL, [8-26](#)

SET_APPLICATION_ID procedure
 APEX_APPLICATION_INSTALL, [8-27](#)

SET_APPLICATION_NAME procedure
 APEX_APPLICATION_ADMIN, [7-13](#)
 APEX_APPLICATION_INSTALL, [8-27](#)

SET_APPLICATION_STATUS procedure
 APEX_APPLICATION_ADMIN, [7-13](#)
 APEX_UTIL, [59-104](#)

SET_APPLICATION_VERSION procedure
 APEX_APPLICATION_ADMIN, [7-15](#)

SET_ARRAY_CURRENT_ROW procedure
 APEX_EXEC, [27-92](#)

SET_ARRAY_ROW_VERSION_CHECKSUM
 procedure
 APEX_EXEC, [27-92](#)

SET_ATTRIBUTE procedure
 APEX_UTIL, [59-106](#)

SET_AUTHENTICATION_RESULT procedure
 APEX_UTIL, [59-107](#)

SET_AUTHENTICATION_SCHEME procedure
 APEX_APPLICATION_ADMIN, [7-15](#)
 APEX_APPLICATION_INSTALL, [8-28](#)

SET_AUTO_INSTALL_SUP_OBJ procedure
 APEX_APPLICATION_INSTALL, [8-29](#)

SET_BUILD_OPTION_STATUS procedure
 APEX_APPLICATION_ADMIN, [7-16](#)
 APEX_UTIL, [59-108](#)

SET_BUILD_STATUS function
 APEX_APPLICATION_INSTALL, [8-30](#)

SET_BUILD_STATUS procedure
 APEX_APPLICATION_ADMIN, [7-17](#)

SET_COMPONENT_VALUES procedure
 APEX_PLUGIN_UTIL, [45-55](#)

SET_CSV_PARAMETERS procedure
 APEX_ESCAPE, [26-20](#)

SET_CURRENT_ROW procedure
 APEX_EXEC, [27-93](#)

SET_CURRENT_STYLE procedure
 APEX_THEME, [57-4](#)

SET_CURRENT_THEME_STYLE procedure
 APEX_UTIL, [59-109](#)

SET_CUSTOM_AUTH_STATUS procedure
 APEX_UTIL, [59-111](#)

SET_DATABASE_CREDENTIAL procedure
 APEX_CREDENTIAL, [16-6](#)

SET_DEBUG procedure
 APEX_SESSION, [51-5](#)

SET_EDITION procedure
 APEX_UTIL, [59-112](#)

SET_EMAIL procedure
 APEX_UTIL, [59-112](#)

SET_FILE_STORAGE procedure
 APEX_APPLICATION_ADMIN, [7-17](#)

SET_FIRST_NAME procedure
 APEX_UTIL, [59-113](#)

SET_GLOBAL_NOTIFICATION procedure
 APEX_APPLICATION_ADMIN, [7-18](#)
 APEX_UTIL, [59-114](#)

SET_GROUP_GROUP_GRANTS procedure
 APEX_UTIL, [59-115](#)

SET_GROUP_USER_GRANTS procedure
 APEX_UTIL, [59-115](#)

SET_HTML_ESCAPING_MODE procedure
 APEX_ESCAPE, [26-21](#)

SET_IMAGE_PREFIX procedure
 APEX_APPLICATION_ADMIN, [7-19](#)
 APEX_APPLICATION_INSTALL, [8-30](#)

SET_INITIATOR_CAN_COMPLETE procedure
 APEX_APPROVAL, [9-30](#)
 APEX_HUMAN_TASK, [31-25](#)

SET_KEEP_BACKGROUND_EXECS procedure
 APEX_APPLICATION_INSTALLf, [8-31](#)

SET_KEEP_SESSIONS procedure
 APEX_APPLICATION_INSTALL, [8-32](#)

SET_LAST_NAME procedure
APEX_UTIL, [59-116](#)

SET_LOG_LEVEL procedure
APEX_WORKFLOW, [61-15](#)

SET_LOG_SWITCH_INTERVAL procedure
APEX_INSTANCE_ADMIN, [32-34](#)

SET_MAX_SCHEDULER_JOBS procedure
APEX_APPLICATION_ADMIN, [7-20](#)
APEX_APPLICATION_INSTALL, [8-33](#)

SET_NULL procedure
APEX_EXEC, [27-93](#)

SET_OFFSET procedure
APEX_APPLICATION_INSTALL, [8-33](#)

SET_PARAMETER procedure
APEX_INSTANCE_ADMIN, [32-35](#)

SET_PARSER_FLAGS procedure
APEX_DATA_PARSER, [22-15](#)

SET_PARSING_SCHEMA procedure
APEX_APPLICATION_ADMIN, [7-20](#)

SET_PARSING_SCHEMA_FOR_REQUEST procedure
APEX_UTIL, [59-117](#)

SET_PASS_ECID procedure
APEX_APPLICATION_ADMIN, [7-21](#)
APEX_APPLICATION_INSTALL, [8-34](#)

SET_PERSISTENT_CREDENTIALS procedure
APEX_CREDENTIAL, [16-7, 16-8](#)

SET_PERSISTENT_CREDENTIALS procedure
signature 2
APEX_CREDENTIAL, [16-8](#)

SET_PERSISTENT_TOKEN procedure
APEX_CREDENTIAL, [16-9](#)

SET_PREFERENCE procedure
APEX_UTIL, [59-118](#)

SET_PROGRESS procedure
APEX_BACKGROUND_PROCESS, [13-5](#)

SET_PROXY procedure
APEX_APPLICATION_INSTALL, [8-35](#)

SET_PROXY_SERVER procedure
APEX_APPLICATION_ADMIN, [7-22](#)

SET_REMOTE_SERVER procedure
APEX_APPLICATION_ADMIN, [7-22](#)
APEX_APPLICATION_INSTALL, [8-36](#)

SET_REQUEST_ECID_CONTEXT procedure
APEX_WEB_SERVICE, [60-30](#)

SET_REQUEST_HEADERS procedure
APEX_WEB_SERVICE, [60-31](#)

SET_REST_SOURCE_CATALOG_GROUP procedure
APEX_APPLICATION_INSTALL, [8-37](#)

SET_ROW_VERSION_CHECKSUM procedure
APEX_EXEC, [27-94](#)

SET_SCHEMA procedure
APEX_APPLICATION_INSTALL, [8-38](#)

SET_SCOPE procedure
APEX_CREDENTIAL, [16-10](#)

SET_SECURITY_GROUP_ID procedure
APEX_UTIL, [59-118, 59-120](#)

SET_SECURITY_HIGH_CONTRAST_OFF procedure
APEX_UTIL, [59-120](#)

SET_SESSION_CREDENTIALS procedure
signature 1
APEX_CREDENTIAL, [16-10](#)

SET_SESSION_CREDENTIALS procedure
signature 2
APEX_CREDENTIAL, [16-11](#)

SET_SESSION_CREDENTIALS procedure
signature 3
APEX_CREDENTIAL, [16-12](#)

SET_SESSION_ID procedure
APEX_CUSTOM_AUTH, [18-12](#)

SET_SESSION_ID_TO_NEXT_VALUE procedure
APEX_CUSTOM_AUTH, [18-12](#)

SET_SESSION_MAX_IDLE_SECONDS procedure
APEX_UTIL, [59-122](#)

SET_SESSION_SCREEN_READER_OFF procedure
APEX_UTIL, [59-123](#)

SET_SESSION_SCREEN_READER_ON procedure
APEX_UTIL, [59-123](#)

SET_SESSION_STATE procedure
APEX_UTIL, [59-121](#)

SET_SESSION_STYLE procedure
APEX_THEME, [57-5](#)

SET_SESSION_STYLE_CSS procedure
APEX_THEME, [57-6](#)

SET_SESSION_TERRITORY procedure
APEX_UTIL, [59-124](#)

SET_SESSION_TIME_ZONE procedure
APEX_UTIL, [59-125](#)

SET_SESSION_TOKEN procedure
APEX_CREDENTIAL, [16-12](#)

SET_STATUS procedure
APEX_BACKGROUND_PROCESS, [13-7](#)

SET_TASK_DUE procedure
APEX_APPROVAL, [9-31](#)
APEX_HUMAN_TASK, [31-25](#)

SET_TASK_PARAMETER_VALUES procedure
APEX_APPROVAL, [9-32](#)
APEX_HUMAN_TASK, [31-26](#)

SET_TASK_PRIORITY procedure
APEX_APPROVAL, [9-33](#)
APEX_HUMAN_TASK, [31-27](#)

SET_TENANT_ID procedure
APEX_SESSION, [51-6](#)

SET_THEME_ID procedure
APEX_APPLICATION_INSTALL, [8-39](#)

SET_TRACE procedure
APEX_SESSION, [51-6](#)

SET_USER procedure
 APEX_CUSTOM_AUTH, [18-12](#)

SET_USER_CONSENT procedure
 APEX_AI, [3-9](#)

SET_USER_STYLE procedure
 APEX_THEME, [57-7](#)

SET_USERNAME procedure
 APEX_UTIL, [59-125](#)

SET_VALUE procedure
 APEX_APP_SETTING, [5-1](#)
 APEX_EXEC, [27-96](#)

SET_VALUE procedure signature 1
 APEX_SESSION_STATE, [52-3](#)

SET_VALUE procedure signature 2
 APEX_SESSION_STATE, [52-3](#)

SET_VALUE procedure signature 3
 APEX_SESSION_STATE, [52-4](#)

SET_VALUES procedure
 APEX_EXEC, [27-100](#)

SET_WORKSPACE procedure
 APEX_APPLICATION_INSTALL, [8-40](#)
 APEX_UTIL, [59-126](#)

SET_WORKSPACE procedure signature 1
 APEX_EXTENSION, [29-3](#)

SET_WORKSPACE procedure signature 2
 APEX_EXTENSION, [29-4](#)

SET_WORKSPACE_CONSUMER_GROUP procedure
 APEX_INSTANCE_ADMIN, [32-36](#)

SET_WORKSPACE_ID procedure
 APEX_APPLICATION_INSTALL, [8-39](#)

SET_WORKSPACE_PARAMETER procedure
 APEX_INSTANCE_ADMIN, [32-36](#)

Setting Cookies and HTTP Headers
 APEX_WEB_SERVICE, [60-5](#)

SHOW_HIGH_CONTRAST_MODE_TOGGLE procedure
 APEX_UTIL, [59-127](#)

SHOW_SCREEN_READER_MODE_TOGGLE procedure
 APEX_UTIL, [59-128](#)

SHUFFLE function
 APEX_STRING, [55-21](#)

SHUFFLE procedure
 APEX_STRING, [55-22](#)

SKIP_CURRENT_ROW procedure
 APEX_AUTOMATION, [12-12](#)

SOAP web service, [60-2](#)

SORT_MEMBERS procedure
 APEX_COLLECTION, [15-28](#)

SPATIAL_IS_AVAILABLE function
 APEX_SPATIAL, [54-7](#)

special characters, encoding, [59-140](#)

SPLIT function signature 1
 APEX_STRING, [55-22](#)

SPLIT function signature 2
 APEX_STRING, [55-23](#)

SPLIT_CLOBS function
 APEX_STRING, [55-24](#)

SPLIT_MULTIPLE_VALUE_TO_TABLE function
 APEX_PLUGIN_UTIL, [45-57](#)

SPLIT_NUMBERS function
 APEX_STRING, [55-24](#)

START_WORKFLOW function
 APEX_WORKFLOW, [61-15](#)

STOP_APEX_ENGINE Procedure
 APEX_APPLICATION, [6-5](#)

STOP_DATA_GENERATION procedure
 APEX_DG_DATA_GEN, [24-23](#)

STRING_TO_TABLE function
 APEX_STRING, [55-25](#)

STRING_TO_TABLE function (deprecated)
 APEX_UTIL, [59-128](#)

STRINGIFY function signature 1
 APEX_JSON, [37-26](#)

STRINGIFY function signature 2
 APEX_JSON, [37-27](#)

STRINGIFY function signature 3
 APEX_JSON, [37-27](#)

STRINGIFY function signature 4
 APEX_JSON, [37-28](#)

STRINGIFY function signature 5
 APEX_JSON, [37-29](#)

STRIPHTML function signature 1
 APEX_ESCAPE, [26-22](#)

STRIPHTML function signature 2
 APEX_ESCAPE, [26-23](#)

STRONG_PASSWORD_CHECK procedure
 APEX_UTIL, [59-129](#)

STRONG_PASSWORD_VALIDATION function
 APEX_UTIL, [59-133](#)

SUBMIT_FEEDBACK procedure
 APEX_UTIL, [59-134](#)

SUBMIT_FEEDBACK_FOLLOWUP procedure
 APEX_UTIL, [59-135](#)

SUBMIT_INFORMATION procedure
 APEX_APPROVAL, [9-33](#)
 APEX_HUMAN_TASK, [31-27](#)

SUBSCRIBE_PUSH_NOTIFICATIONS procedure
 APEX_PWA, [47-3](#)

SUSPEND procedure
 APEX_WORKFLOW, [61-16](#)

SUSPEND_BACKGROUND_EXECS procedure
 APEX_APPLICATION_INSTALL, [8-40](#)

SWITCH function
 APEX_ITEM, [35-22](#)

SYNCH_TABLE procedure
 APEX_UI_DEFAULT_UPDATE, [58-7](#)

SYNCHRONIZE_DATA procedure
 APEX_REST_SOURCE_SYNC, [49-6](#)

`SYNCHRONIZE_TABLE_DEFINITION` procedure
`APEX_REST_SOURCE_SYNC`, 49-7

T

- `t_dir_entries`
 - `APEX_ZIP`, 62-1
- `t_dir_entry`
 - `APEX_ZIP`, 62-1
- `t_files`
 - `APEX_ZIP`, 62-1
- `t_token`
 - `APEX_JWT`, 38-1
- `TABLE_TO_CLOB` function
 - `APEX_STRING`, 55-25
- `TABLE_TO_STRING` function
 - `APEX_STRING`, 55-26
- `TABLE_TO_STRING` function (deprecated)
 - `APEX_UTIL`, 59-136
- `TERMINATE` procedure
 - `APEX_AUTOMATION`, 12-12
 - `APEX_WORKFLOW`, 61-17
- `TERMINATE` procedure signature 1
 - `APEX_BACKGROUND_PROCESS`, 13-7
- `TERMINATE` procedure signature 2
 - `APEX_BACKGROUND_PROCESS`, 13-8
- `TERMINATE_FAULTED_WORKFLOWS`
 - procedure
 - `APEX_WORKFLOW`, 61-17
- `TEXT` function
 - `APEX_ITEM`, 35-23
- `TEXT_FROM_LOV` function
 - `APEX_ITEM`, 35-25
- `TEXT_FROM_LOV_QUERY` function
 - `APEX_ITEM`, 35-26
- `TEXTAREA` function
 - `APEX_ITEM`, 35-24
- `TO_DISPLAY_FILESIZE` function
 - `APEX_STRING_UTIL`, 56-11
- `TO_HTML` function
 - `APEX_MARKDOWN`, 42-1
- `TO_MEMBER_NAME` function
 - `APEX_JSON`, 37-29
- `TO_XMLTYPE` function
 - `APEX_JSON`, 37-30
- `TO_XMLTYPE_SQL` function
 - `APEX_JSON`, 37-31
- `TOCHAR` function
 - `APEX_DEBUG`, 23-17
- `TRACE` procedure
 - `APEX_DEBUG`, 23-18
- `TRUNCATE_COLLECTION` procedure
 - `APEX_COLLECTION`, 15-29
- `TRUNCATE_LOG` procedure
 - `APEX_INSTANCE_ADMIN`, 32-38

U

- `UNEXPIRE_END_USER_ACCOUNT` procedure
 - `APEX_UTIL`, 59-137
- `UNEXPIRE_WORKSPACE_ACCOUNT`
 - procedure
 - `APEX_UTIL`, 59-138
- `UNLOCK_USER` procedure
 - `APEX_INSTANCE_ADMIN`, 32-39
- `UNRESTRICT_SCHEMA` procedure
 - `APEX_INSTANCE_ADMIN`, 32-39
- `UNSUBSCRIBE_PUSH_NOTIFICATIONS`
 - procedure
 - `APEX_PWA`, 47-4
- `UNZIP` function
 - `APEX_EXPORT`, 28-6
- `UPD_AD_COLUMN` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-7
- `UPD_AD_SYNONYM` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-8
- `UPD_COLUMN` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-9
- `UPD_DISPLAY_IN_FORM` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-11
- `UPD_DISPLAY_IN_REPORT` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-11
- `UPD_FORM_REGION_TITLE` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-12
- `UPD_GROUP` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-13
- `UPD_ITEM_DISPLAY_HEIGHT` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-14
- `UPD_ITEM_DISPLAY_WIDTH` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-14
- `UPD_ITEM_FORMAT_MASK` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-15
- `UPD_ITEM_HELP` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-16
- `UPD_ITEM_LABEL` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-16
- `UPD_REPORT_ALIGNMENT` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-17
- `UPD_REPORT_FORMAT_MASK` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-17
- `UPD_REPORT_REGION_TITLE` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-18
- `UPD_TABLE` procedure
 - `APEX_UI_DEFAULT_UPDATE`, 58-19
- `UPDATE_BLUEPRINT` procedure
 - `APEX_DG_DATA_GEN`, 24-24
- `UPDATE_COLUMN` procedure
 - `APEX_DG_DATA_GEN`, 24-24
- `UPDATE_DATA_SOURCE` procedure
 - `APEX_DG_DATA_GEN`, 24-28
- `UPDATE_LANGUAGE_MAPPING` procedure
 - `APEX_LANG`, 39-13

UPDATE_MEMBER procedure
 APEX_COLLECTION, [15-30](#)

UPDATE_MEMBER_ATTRIBUTE procedure
 signature 1
 APEX_COLLECTION, [15-33](#)

UPDATE_MEMBER_ATTRIBUTE procedure
 signature 2
 APEX_COLLECTION, [15-34](#)

UPDATE_MEMBER_ATTRIBUTE procedure
 signature 3
 APEX_COLLECTION, [15-35](#)

UPDATE_MEMBER_ATTRIBUTE procedure
 signature 4
 APEX_COLLECTION, [15-36](#)

UPDATE_MEMBER_ATTRIBUTE procedure
 signature 5
 APEX_COLLECTION, [15-37](#)

UPDATE_MEMBER_ATTRIBUTE procedure
 signature 6
 APEX_COLLECTION, [15-38](#)

UPDATE_MEMBERS procedure
 APEX_COLLECTION, [15-31](#)

UPDATE_MESSAGE procedure signature 1
 APEX_LANG, [39-15](#)

UPDATE_MESSAGE procedure signature 2
 APEX_LANG, [39-16](#)

UPDATE_TABLE procedure
 APEX_DG_DATA_GEN, [24-29](#)

UPDATE_TRANSLATED_STRING procedure
 APEX_LANG, [39-17](#)

UPDATE_VARIABLES procedure
 APEX_WORKFLOW, [61-18](#)

UPLOAD_TEMPLATE function
 APEX_PRINT, [46-6](#)

URL_ENCODE function
 APEX_UTIL, [59-140](#)

user
 get e-mail address, [59-53](#)
 remove preference, [59-96](#)

user account
 altering, [59-28](#)
 creating new, [59-15](#)
 fetching, [59-36](#), [59-39](#), [59-41](#)
 removing, [59-97](#), [59-98](#)
 update email address, [59-112](#)
 updating FIRST_NAME, [59-113](#)
 updating LAST_NAME value, [59-116](#)
 updating USER_NAME value, [59-125](#)

V

VALIDATE procedure
 APEX_JWT, [38-4](#)

VALIDATE_BLUEPRINT procedure
 APEX_DG_DATA_GEN, [24-30](#)

VALIDATE_EMAIL_CONFIG Procedure
 APEX_INSTANCE_ADMIN, [32-40](#)

VALIDATE_INSTANCE_SETTING procedure
 APEX_DG_DATA_GEN, [24-30](#)

variables
 APEX_APPLICATION, [6-3](#)
 APEX_WEB_SERVICE, [60-7](#)

W

WARN procedure
 APEX_DEBUG, [23-19](#)

Web Credentials
 APEX_WEB_SERVICE, [60-6](#)

workspace
 export file, [59-34](#)
 numeric security group ID, [59-44](#)

WORKSPACE_ACCOUNT_DAYS_LEFT function
 APEX_UTIL, [59-141](#)

WRITE procedure
 APEX_JSON, [37-44](#)

WRITE procedure signature 1
 APEX_JSON, [37-32](#)

WRITE procedure signature 10
 APEX_JSON, [37-37](#)

WRITE procedure signature 11
 APEX_JSON, [37-37](#)

WRITE procedure signature 12
 APEX_JSON, [37-38](#)

WRITE procedure signature 13
 APEX_JSON, [37-38](#)

WRITE procedure signature 14
 APEX_JSON, [37-39](#)

WRITE procedure signature 15
 APEX_JSON, [37-40](#)

WRITE procedure signature 16
 APEX_JSON, [37-40](#)

WRITE procedure signature 17
 APEX_JSON, [37-41](#)

WRITE procedure signature 18
 APEX_JSON, [37-42](#)

WRITE procedure signature 19
 APEX_JSON, [37-42](#)

WRITE procedure signature 2
 APEX_JSON, [37-33](#)

WRITE procedure signature 20
 APEX_JSON, [37-43](#)

WRITE procedure signature 3
 APEX_JSON, [37-33](#)

WRITE procedure signature 4
 APEX_JSON, [37-33](#)

WRITE procedure signature 5
 APEX_JSON, [37-34](#)

WRITE procedure signature 6
 APEX_JSON, [37-34](#)

WRITE procedure signature 7
 APEX_JSON, [37-35](#)
WRITE Procedure Signature 8
 APEX_JSON, [37-36](#)
WRITE procedure signature 9
 APEX_JSON, [37-36](#)
WRITE_CONTEXT procedure
 APEX_JSON, [37-44](#)

Z

ZIP function
 APEX_EXPORT, [28-7](#)