Working with the APEX_ODA PLSQL Package

Sydney Nurse, July 2021

This paper attempts to describe the APEX_ODA PLSQL package and procedures and how-to implement them into an APEX application. The code sample provided can be modified to suite the readers use case as a template for accessing Rest API programmatically for APEX applications.

The sample Code provided are not explained in depth (line-by-line). It is assumed the reader has some APEX or PLSQL experience and seeking to integrate with Oracle Digital Assistant(ODA) using Oracle APEX/Database features.

The outbound integration goal is to update ODA dynamic entities as the data is managed within the database application. i.e. Insert, Update, Delete operations

In brief Dynamic Entities are value list which are managed through the ODA Rest APIs. They are similar to a key-value pair list having a value with/without synonyms. As of the writing of this document, ODA version 21.06 is the current release.

This article does not delve into ODA Entity types or Dynamic Entities, those topics are covered in depth by previously release TechExchange articles. It is suggested that the Dynamic Entities in Oracle Digital Assistant, by Frank Nimphius & Chris Kutler from November 2019 be reviewed prior to this article as it provides core information regarding Dynamic Entities and the required Rest APIs.

A note about the author I am a dabbler, curious about technology but certainly not a coder by trade or Oracle PLSQL developer. I focus on function not style or elegance, feel free to add this to your own variant, my goal was to have a working integration, that I could reuse, i.e. solely for my purposes.

INDEX

ODA Dynamic Entity Rest API Sequence

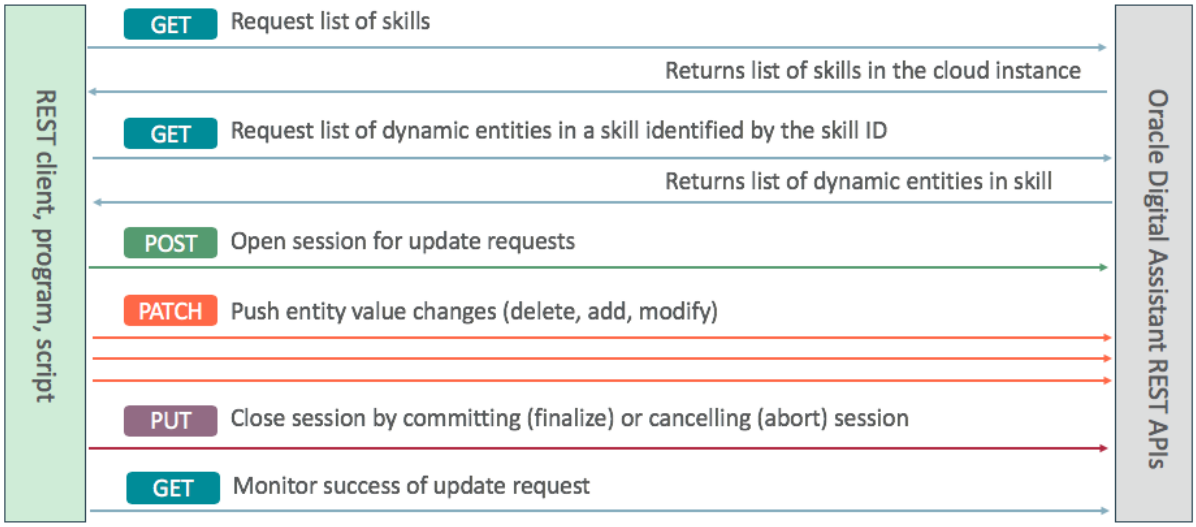

Figure 1: ODA Rest Call Sequence

The order sequence has been implemented by the APEX_ODA PLSQL package with the exception of the last API call reference to monitor the status of the update request.

The APEX_ODA PLSQL Package Specification

The package uses 4 package level variables:

ODA API Required variables

- G_OCI_WEB_CREDENTIALS
- G_ODA_BASE_URL
- G_ODA_SKILL_ID
- G_ODA_DYNAMIC_ENTITY_ID

And five defined procedures:

- PROCEDURE SET_ODA_CONN(l_credential varchar2, l_api_base_url varchar2);
- PROCEDURE GET_SKILL(l_skill_name in varchar2, l_curr_oda_skill_id out varchar2);
- PROCEDURE GET_DYNAMIC_ENTITY(l_dynamic_entity_name in varchar2, l_curr_oda_dynamic_entity_id out varchar2);
- PROCEDURE PUSH_DYNAMIC_ENTITY_DATA(l_push_type varchar2, l_canonicalName varchar2, l_synonyms varchar2);
- PROCEDURE REFRESH_PUSH_DYNAMIC_ENTITY_DATA(l_dynamic_entity_name in varchar2);

To implement the ODA Dynamic Entity Rest API Sequence.

It is suggested to create at a minimum two Application Substitutions, referenceable throughout the application:

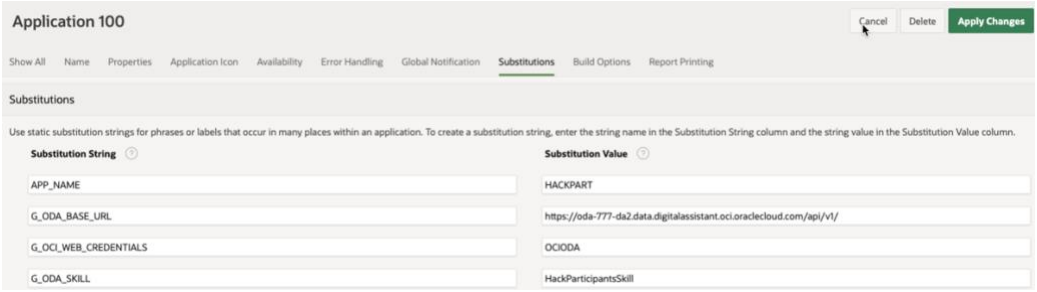

Figure 2: APEX Application Substitutions

Substitution	Description	Example
G_ODA_BASE_URL	The base URL to access the Oracle Digital Assistant instance APIs	https://oda-777-da2.data.digitalassistant.oci.oraclecloud.com/api/v1/
G_OCI_WEB_CREDENTIALS	The static ID of the Web Credential used for Oracle Cloud Infrastructure (OCI) Rest API calls.	OCIODA
G_ODA_SKILL	Optional: For cases when the APEX application will reference a single ODA skill, the name of the ODA Skill	HackParticipantSkill

The APEX_ODA PLSQL Package Procedures

SET_ODA_CONN

This procedure sets the global variables related to ODA connections and accepts two parameters for the OCI Web Credential and the ODA Rest API URL base. When used the global substitution variables it can be invoked as follows:

```
BEGIN
   APEX_ODA.SET_ODA_CONN(:G_OCI_WEB_CREDENTIALS, :G_ODA_BASE_URL);
END;
```

```
-- ==============================================
-- Procedure:      SET_ODA_CONN
-- Description: Sets the Global Variables for the ODA Connection
-- Parameters:
--      @l_credential              - Used to set  @G_OCI_WEB_CREDENTIALS with the OCI connection Credentials
--      @l_api_base_url            - Used to set  @G_ODA_BASE_URL with the ODA connection Base API URL
-- Returns:
-- ==============================================
   PROCEDURE SET_ODA_CONN(l_credential varchar2, l_api_base_url varchar2);
```

GET_SKILL

This procedure invokes the ODA List Skills API, retrieves the Skill Identifier and sets the global variable related to ODA Skill. It accepts a single parameter for the ODA Skill name and returns the Skill ID with the highest version. When used the global substitution variables it can be invoked as follows:

```
DECLARE

   l_skill_id varchar2(4000);
BEGIN
   APEX_ODA.SET_ODA_CONN(:G_OCI_WEB_CREDENTIALS, :G_ODA_BASE_URL);
   APEX_ODA.GET_SKILL(:G_ODA_SKILL, l_skill_id);
END;
```

```
-- ==============================================
-- Procedure:      GET_SKILL
-- Description: Connects to ODA to retrieve the Skill ID for the lastest and highers version of the Skill Name supplied
--              and sets @G_ODA_SKILL_ID with the retrieved Skill ID
-- Parameters:
--      @l_skill_name              - The ODA Skill Name
-- Returns: @l_curr_oda_skill_id    - The ODA Skill ID
-- ==============================================
   PROCEDURE GET_SKILL(l_skill_name in varchar2, l_curr_oda_skill_id out varchar2);
```

GET_DYNAMIC_ENTITY

This procedure invokes the ODA Get Dynamic Entities API, retrieves the Dynamic Entity Identifier and sets the global variable related to ODA Dynamic Entity. It references the global package variables for the connection and ODA Skill and accepts a single parameter for the ODA Dynamic Entity name and returns the Dynamic Entity ID. It can be invoked as follows:

```
DECLARE

   l_skill_id varchar2(4000);
   l_dynamic_entity_id varchar2(4000);
BEGIN
   APEX_ODA.SET_ODA_CONN(:G_OCI_WEB_CREDENTIALS, :G_ODA_BASE_URL);

   APEX_ODA.GET_SKILL(:G_ODA_SKILL, l_skill_id);
```

```
    APEX_ODA.GET_DYNAMIC_ENTITY('hack_skills', l_dynamic_entity_id);
END;
```

```
-- ============================================
-- Procedure:      GET_DYNAMIC_ENTITY
-- Description: Connects to ODA to retrieve the Dynamic Entity ID for the Skill Name supplied
--              and sets @G_ODA_DYNAMIC_ENTITY_ID with the retrieved Dynamic Entity ID
-- Parameters:
--      @l_dynamic_entity_name              - The ODA Dynamic Entity Name
-- Returns: @l_curr_oda_dynamic_entity_id    - The ODA Dynamic Entity ID
-- ============================================
    PROCEDURE GET_DYNAMIC_ENTITY(l_dynamic_entity_name in varchar2, l_curr_oda_dynamic_entity_id out varchar2);
```

PUSH_DYNAMIC_ENTITY_DATA

This procedure implements the API sequence to initialize a push request, submit data for that request, and finalize that request. It follows the "happy path" and does not handle errors for the sequence. It references the global package variables for the connection, ODA Skill and Dynamic Entity. It accepts three parameters for the action or submission type, the canonical name of the value and any associated synonyms as a string array using the colon (':') as a separator and can be invoked from a Page as follows:

```
DECLARE

    l_skill_id varchar2(4000);
    l_dynamic_entity_id varchar2(4000);
BEGIN
    APEX_ODA.SET_ODA_CONN(:G_OCI_WEB_CREDENTIALS, :G_ODA_BASE_URL);

    APEX_ODA.GET_SKILL(:G_ODA_SKILL, l_skill_id);
    APEX_ODA.GET_DYNAMIC_ENTITY('hack_skills', l_dynamic_entity_id);
    APEX_ODA.PUSH_DYNAMIC_ENTITY_DATA(
        case :REQUEST -- Button Request variable
            when 'CREATE' then 'add'
            when 'SAVE' then 'modify'
            when 'DELETE' then 'delete' end,
        :P7_NAME, -- Page variable
        NULL
    );

END;
```

```
-- ============================================
-- Procedure:      PUSH_DYNAMIC_ENTITY_DATA
-- Description: Connects to ODA, manages the Push Data Request for a single Dynamic Entity Value for the Skill & Dynamic Entity Names supplied
-- Parameters:
--      @l_push_type           - The ODA Push Data Request request type: ['add','delete','modify']
--      @l_canonicalName        - The Dynamic Entity Value in the ODA value list
--      @l_synonyms            - A varchar2 array of strings for Dynamic Entity Value Synonyms that refer to the Value in the ODA Value list
-- Returns:
-- ============================================
    PROCEDURE PUSH_DYNAMIC_ENTITY_DATA(l_push_type varchar2, l_canonicalName varchar2, l_synonyms varchar2);
```

REFRESH_PUSH_DYNAMIC_ENTITY_DATA

This procedure is similar to the PUSH_DYNAMIC_ENTITY_DATA procedure, implementing the API sequence to initialize a push request, submit data for that request, and finalize that request. It also follows the "happy path" and does not handle errors for the sequence. It references the global package variables for the connection and ODA Skill. It accepts a single parameter for the Dynamic Entity name. This package is application specific and has been left as a refence implementation on how-to completely refresh an ODA Dynamic Entity from base table objects.

The procedure uses the input parameter to switch which dynamic SQL is executed. The SQL is each case is specific to the dynamic entity and related base table, some with synonyms and other not. The SQL uses the json object and table feature of the database to construct the full json body text submitted in the ODA Push Data Request. It can be invoked from a Page as follows:

```
DECLARE

    l_skill_id varchar2(4000);
BEGIN
    APEX_ODA.SET_ODA_CONN(:G_OCI_WEB_CREDENTIALS, :G_ODA_BASE_URL);

    APEX_ODA.GET_SKILL(:G_ODA_SKILL, l_skill_id);
    APEX_ODA.REFRESH_PUSH_DYNAMIC_ENTITY_DATA('hack_skills');

END;
```

```
-- ============================================
-- Procedure:      REFRESH_PUSH_DYNAMIC_ENTITY_DATA
-- Description: Connects to ODA, manages the Push Data Request to refresh a Dynamic Entity Value list for the Skill & Dynamic Entity Names supplied
--              **This was written specifically to match the Tables to Dynamic Entities of the Hack Participants APEX to ODA app**
-- Parameters:
--      @l_dynamic_entity_name  - The ODA Push Dynamic Entity Name: ['hack_challenges','hack_teams','hack_participants','hack_skills','hack_countries']
-- Returns:
-- ============================================
    PROCEDURE REFRESH_PUSH_DYNAMIC_ENTITY_DATA(l_dynamic_entity_name in varchar2);
```

Using JSON_TABLE to extract data from Rest Responses

Rest API responses can be parsed using several methods, the GET_SKILL and GET_DYNAMIC_ENTITY use the JSON_TABLE to extract the identifier from the response. In the case of the Skill ID, versioning would require the user to provide a specific version to use and loop over the response results or similarly loop to get and compare the version number.

The JSON_TABLE provides SQL access to the json response, for example retrieving the highest version of the requested skill:

```
  -- Using JSON_Table to extract Skill ID and version for the latest version of the skill
  select id,version into l_curr_oda_skill_id, l_curr_oda_skill_version from (
     select id,name,version from json_table(l_response, '$.items[*]'
     columns(
        id      varchar2(2000) path '$.id',
        name    varchar2(2000) path '$.name',
        version varchar2(2000) path '$.version'
        )
     ) jt
     order by version desc
  ) where rownum = 1;
```

Or getting the Dynamic Entity ID and Name:

```
  -- Get the ID for the Dynamic Entity by its name passed in as a parameter => DYNAMIC_ENTITY_NAME
  select id, name  into l_curr_oda_dynamic_entity_id, l_curr_oda_dynamic_entity_name
  from json_table(l_response, '$.items[*]'
     columns(
        id      varchar2(2000) path '$.id',
        name    varchar2(2000) path '$.name'
        )
     ) jt
  where upper(name) = upper(l_dynamic_entity_name ) ;
```

Using JSON_OBJECT to construct json payload for the ODA Push Data to Request API

The package procedure REFRESH_PUSH_DYNAMIC_ENTITY uses dynamic sql based on JSON_OBJECT and JSON_ARRAY.

The JSON_OBJECT contains any value key pairs and for arrays, the JSON_ARRAY is utilized. The text constructed in the procedure handles the use of single quotes used to identify fields and values.

Here are a few raw samples:

SQL Statement	Results
select json_object('add' value json_arrayagg(json_object('canonicalName' value title, 'synonyms' value json_array(label), 'nativeLanguageTag' value 'en')) returning varchar2) challenges from content where type = 'challenges';	{"add":[{"canonicalName":"Quality Education","synonyms":["Education"],"nativeLanguageTag":"en"},{"canonicalName":"Decent Work and Economic Growth","synonyms":["Economic"],"nativeLanguageTag":"en"},{"canonicalName":"Good Health & Wellbeing","synonyms":["Health"],"nativeLanguageTag":"en"}]}
select json_object('add' value json_arrayagg(json_object('canonicalName' value p.first_name || ' ' || p.last_name, 'synonyms' value json_array(c.region || ':' || c.name), 'nativeLanguageTag' value 'en')) returning varchar2) participants from participants p, countries c where c.id = p.country;	{"add":[{"canonicalName":"Martine Richmann","synonyms":["Europe:Switzerland"],"nativeLanguageTag":"en"},{"canonicalName":"Grant Roland","synonyms":["Europe:United Kingdom"],"nativeLanguageTag":"en"}]}
select json_object('add' value json_arrayagg(json_object('canonicalName' value nvl(region,'Other'),'synonyms' value json_array(LISTAGG(name, ':') WITHIN GROUP (ORDER BY name asc)),'nativeLanguageTag' value 'en')) returning varchar2) regions from countries GROUP BY nvl(region,'Other')	{"add":[{"canonicalName":"Americas","synonyms":["French Guiana"],"nativeLanguageTag":"en"},{"canonicalName":"Oceania","synonyms":["French Polynesia"],"nativeLanguageTag":"en"}]}

The examples provided are formatted as required by ODA for the Dynamic Entity Push Data Response. The full refresh only uses the 'add' action type and forces the request to overwrite any existing data (not shown).

Take care when translating the proven query into the text variable that stores it, managing quotes and concatenation correctly.

Summary

In summary, the APEX_ODA PLSQL package can be imported into any Oracle Database that has the APEX package libraries. It has been designed specifically for APEX applications requiring an outbound (push) integration to update ODA Dynamic Entities that are managed via Restful APIs.

The sample package code has been provided free of any changes and limitations in hope that it can foster better and faster development projects in the future.