**APEX Demonstration Script** 

**Integrating APEX with the Oracle Developer Cloud Service**

*v1.2 | Dec 2015 | Jules Lane, Ewan Slater, Edi Vasermann*

*-----------------------------------------------------------------------------------------------------------------------------------------*

**Background**

APEX 5 is the fastest and easiest way to develop web and mobile apps, either in the cloud or on premise. The Developer Cloud Service is integrated with the Oracle Database Cloud Service and provides a comprehensive environment which centralizes and automates the management, control and deployment of applications of all types.

This document describes a demonstration showing how DevCS can complement Oracle Application Express (APEX) development by automating the management, provisioning and continuous integration of APEX apps or components. APEX developers benefit from controlled but automated development and testing and I.T. architects benefit from inclusion of APEX in standard corporate development control processes

**Sponsors**

This project has the full support of APEX product management, Database Cloud product management, EMEA Cloud Architects and APEX partners who are keen to beta test. We also hope to gain the support of Dev CS Cloud product management.

**Environments**

This demonstration assumes the use of APEX 5.0.1 in source and target DBaaS environments.

**Product Enhancements**

It is hoped that this demonstration can form the basis of appropriate product developments to at least partially automate steps 3 - 7 of the process below.

**Process Flow**

1. Run (or build and run) simple app in any APEX environment
2. Export app components to sql and xml files
3. Copy files to local storage (local GIT repository)
4. Upload files into the DevCS GIT repository
5. Run pre-defined set of Hudson jobs to;
   1. Execute a script to provision a new DBaaS environment
   2. Execute a database script to import the sql and XML files into the APEX environment in the new DBaaS service in order to create the schema objects and APEX app.
6. Run the APEX app in the new DBaaS APEX environment.
7. Automate a pre-defined testing process

**Step 1. Build basic prototype**

See other APEX Demo scripts

**Step 2. Export Components**

Using standard APEX tools;

* Export Workspace to sql file
* Export Application to sql file
* Generate DDL for 3 tables and related schema objects and save as a sql file
* Download data from the 3 tables as XML

**Step 3. Load files into DevCS GIT repository**

*ES to define process*

**Step 4. Run Hudson job to execute a script to provision a new DBaaS environment**

*ES to define process*

**Step 5. Run Hudson job to load APEX app into the new DBaaS environment**

Execution sequence should be;

1. Create Schema
2. Workspace.sql
3. SchemaObjects.sql
4. App.sql
5. Load XML data files

*Schema creation*

Before running these scripts you will need to create a schema with the same name as the workspace (for this demo its JULES). You can get this from the name of the Workspace file, or *more reliably* from the <p\_ws\_schema => 'JULES'> parameter within that file ).

*User ID’s*  
To run the first script (to import the workspace and APEX users) use an id with DBA privileges like SYSDBA.   
  
To run the others use SYSDBA, any user with the APEX\_ADMINISTRATOR\_ROLE role (e.g. APEX\_050000) or the user created during workspace creation (JULES in this example).

*ES to define Hudson process to execute the script*

**Step 6. Run the app in the new DBaaS environment**

* Login to APEX in the DBaaS environment manually as user created in step 5.
* Run the App
* OR manually run the app from the url

*Application url*

Examples of APEX app url are;

* localhost:8080/apex/f?p=104:1.
  + Means run the APEX app id 104, page 1.
* <https://apex.oracle.com/pls/apex/f?p=50503:3>
* *https://databasetrial2534-gboracleuktrial42823.db.us2.oraclecloudapps.com/apex/f?p=10800041:1*

The second part should be the same (/apex/f?p=<app\_id>:<page\_number>) in the new DBaaS environment as it was in the source APEX environment.  
The script will need to concatenate this to the correct first part ('server name:ORDS port').

**Step 7. Automate execution and testing of the app**

* Run the app from the url within a testing environment (*ES to define*)