A Guide for implementing Oracle Hyperion Essbase Custom Defined Functions

Challenge: Many times when developing Essbase calculations or Hyperion Planning Business Rules, the out-of-box functions provided are useful for providing static type calculations based on conditional logic. This is good for creating rich analytic applications with simple logic and mathematics that can accurately represent the data. When it comes to complex calculations with dynamically changing conditions or using iterative methods, an Essbase CDF is the solution. Extending Essbase calculation scripts to include custom functions can add a layer of analytic or forecasting metrics to existing legacy financial data. A CDF could also be used to update existing data from web services for example, currency rates in a Planning application. CDF's are a great solution for integrating into existing Planning, Budgeting and Forecasting Cloud Services (PBCS).

Essbase contains a rich function set for building calculation scripts; however the calculations can be extended to include any number or type of computations or algorithms. This Guide will review the basic steps to get started. Other more detailed guides may follow on implementing additional specific application examples.

Applications of the CDF:

- Goal Seeking
- Dynamic outline building
- Time Encoding/Decoding
- Live Data Feed
- Computational Algorithms and Methods
- Integration with Essbase API.
- Integration with other languages and protocols such as R script, Jython, Python, Ruby, and Email.
- Anonymous Lambda Functions for dynamic programming http://en.wikipedia.org/wiki/Anonymous_function
- Anything else that Essbase can't do easily
- Built with Java so the potential is unlimited.

Solution: Use the following guide to create an Essbase custom defined function or CDF. This guide uses the simplest example also provided by many other instructional to create a new function that sums a list of values. Use the simplest example to setup you development environment and become familiar with the steps to implement CDF's.

Step 1 - Develop

- Design and build Java function classes.
- Compile using a matching JDK (list of Essbase version and Java required)
- Package function classes into a jar library
- Deploy the function library jar to Essbase

Step 2 - Deploy and Register

- Define the function profile and data interface @JCustomFunction(params...in/out)

- Define security policy
- Create a registration script
- Register functions into a LOCAL database for testing Calculation Script

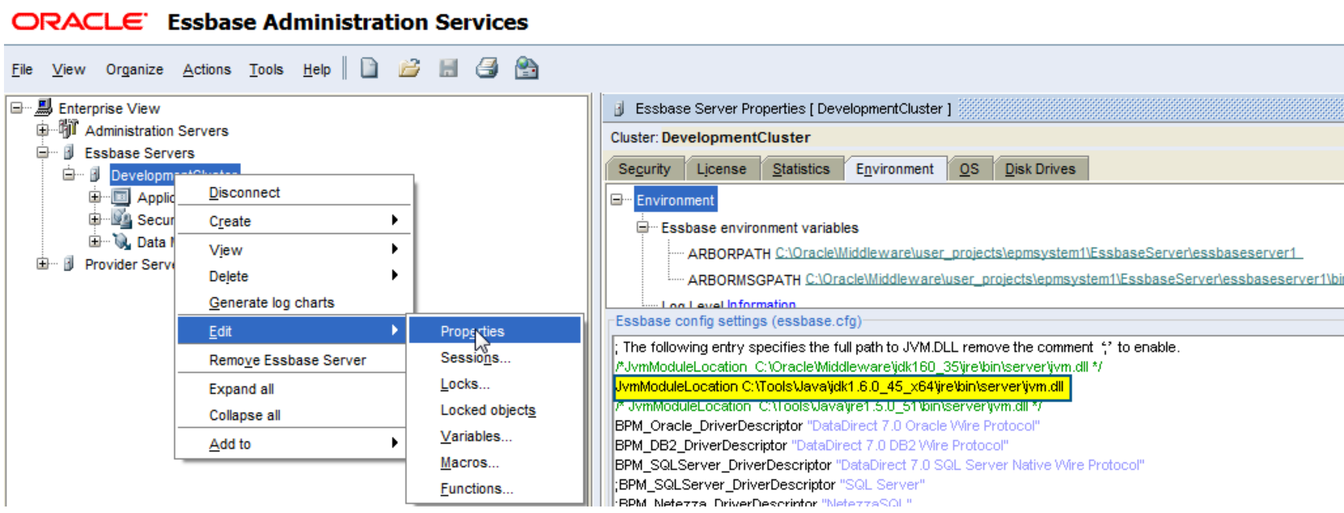
Step 3 - Calculation Script

- Develop Essbase calculation scripts or business rules using new external functions
- Pass data from your intersections to the new @JCustomFunction()
- Return values can be assigned to Essbase members or used as input to other functions.

Getting Started with Essbase Custom Defined Functions CDF's

Before starting, review the requirements and gather the required tools and documents. Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers and allows you to compile the Java code into a deployable java archive (JAR).

1. You will need to have a matching version of the Java Development Kit (JDK) 32bit or 64bit http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Text editor http://notepad-plus-plus.org/
    a. Java Integrated Development Environment (IDE) JDeveloper, Eclipse, or NetBeans
3. Enable Java for Essbase in the Essbase.cfg or use the Essbase Administration Console
    a. ; The following entry specifies the full path to JVM.DLL remove the comment ';' to enable.
    JvmModuleLocation {path}\jdk{version}\jre\bin\server\jvm.dll



```
ARBORPATH C:\Oracle\Middleware\user_projects\epmsystem1\EssbaseServer\essbaseserver1
ARBORMSGPATH C:\Oracle\Middleware\user_projects\epmsystem1\EssbaseServer\essbaseserver1\bi
```

```
; The following entry specifies the full path to JVM.DLL remove the comment ';' to enable.
/*JvmModuleLocation  C:\Oracle\Middleware\jdk160_35\jre\bin\server\jvm.dll */
JvmModuleLocation C:\Tools\Java\jdk1.6.0_45_x64\jre\bin\server\jvm.dll
/* JvmModuleLocation  C:\Tools\Java\jre1.5.0_51\bin\server\jvm.dll */
BPM_Oracle_DriverDescriptor "DataDirect 7.0 Oracle Wire Protocol"
BPM_DB2_DriverDescriptor "DataDirect 7.0 DB2 Wire Protocol"
BPM_SQLServer_DriverDescriptor "DataDirect 7.0 SQL Server Native Wire Protocol"
;BPM_SQLServer_DriverDescriptor "SQL Server"
;BPM_Netezza_DriverDescriptor "NetezzaSQL"
```

4. Access the Essbase server environment for updating security policy and deploying function package.
5. Documentation and Resources – Oracle Sample CDF's, Essbase DBAG, Oracle TechNet, Java
    a. http://docs.oracle.com/cd/E12032_01/doc/epm.921/html_techref/maxl/ddl/cdf/cdfindex.htm
    b. http://docs.oracle.com/cd/E12825_01/epm.111/esb_dbag/frameset.htm?dcaudfs.htm

Step 1 – Develop functions using Java Development Kit

Write a public Java class with a “public static: method to be used as a custom-defined function. The Essbase Java methods can have any of the following supported data types as input parameters or return values. Returned data types can be void or are converted to Essbase data types.

Java type	Maps to Essbase type
boolean	com.hyperion.essbase.calculator.CalcBoolean
byte	double
char	double
short	double
int	double
long	double
float	double
double	double
java.lang.String	Essbase String Text
com.hyperion.essbase.calculator.CalcBoolean	TRUE, FALSE, and #MISSING
com.hyperion.essbase.calculator.MISSING	#MISSING
Array types[]	

* Doubles need to be checked for infinite or NAN (Not-a-Number) and return finite values or #MISSING instead.

Create a file called FunctionLibrary.java using a text editor or Java IDE and copy the code below. Save the file and open a command prompt.

Code: FunctionLibrary.java

```
public class FunctionLibrary {
  public static double SumData (double[] essbasedata) {
     int dataPosition, dataLength = essbasedata.length;
     double sum = 0.0d;
     if(dataLength == 0) return com.hyperion.essbase.calculator.MISSING;
     for (dataPosition =0; dataPosition < dataLength; dataPosition ++) {
         double dataCell = essbasedata [dataPosition];
         sum += dataCell;
      }
     return sum;
   }
}
```

Compile the Java class using the matching Java javac tool. This tool is found in the jdk/bin/javac and can be used at the command prompt. Move to the directory containing the java file you want to compile and use the javac tool. For example, to compile the FunctionLibrary.java into a class file, use the following command:

Command: javac

```
>javac FunctionLibrary.java
```

Resolve all compiling errors and until the javac compiler creates a new .class file, for example FunctionLibrary.class. Once the class file is created, it needs to be packaged into a deployable archive.

Package the new class file(s) into a deployable Java archive jar file. Once the jar file is created, to needs to be deployed and registered with Essbase.

Command: jar

```
>jar cf FunctionLibrary.jar FunctionLibrary.class
```

Step 2 – Deploy and Register functions with Essbase

The next step is to copy the newly created jar file into the Essbase server location and update the security policy. CDF packages are located {MIDDLEWARE_HOME}\EPMSystem11R1\products\Essbase\EssbaseServer\java\udf.   Modify the udf.policy file and give the new java package permission. The Essbase server will need to be restarted in order to load the new jar file. If the file is being replaced with a new version, then the Essbase server may have to be stopped while doing the copy step.

Command: copy

```
>copy FunctionLibrary.jar
{MIDDLEWARE_HOME}\EPMSystem11R1\products\Essbase\EssbaseServer\java\udf
```

entry: udf.policy

```
// Grant all permissions to CDF's
grant codeBase "file:${essbase.java.home}/udf/FunctionLibrary.jar" {
  permission java.security.AllPermission;
};
```

Command: maxl registration

```
MAXL> create function Sample.'@JSumData '
  2> as 'FunctionLibrary.SumData'
  3> spec '@JSumData(memberRange)
  4> comment 'A sample CDF functions that sums data';
```

Once the Essbase server is restarted check the Essbase application level logs to verify the new library and function have been loaded.

```
[Mon Feb 10 14:57:46 2014]Local/Sample///7484/Info(1002035)

Starting Essbase Server - Application [Sample]

Loaded and initialized JVM module

External [LOCAL] function [@JSumData] registered OK
```

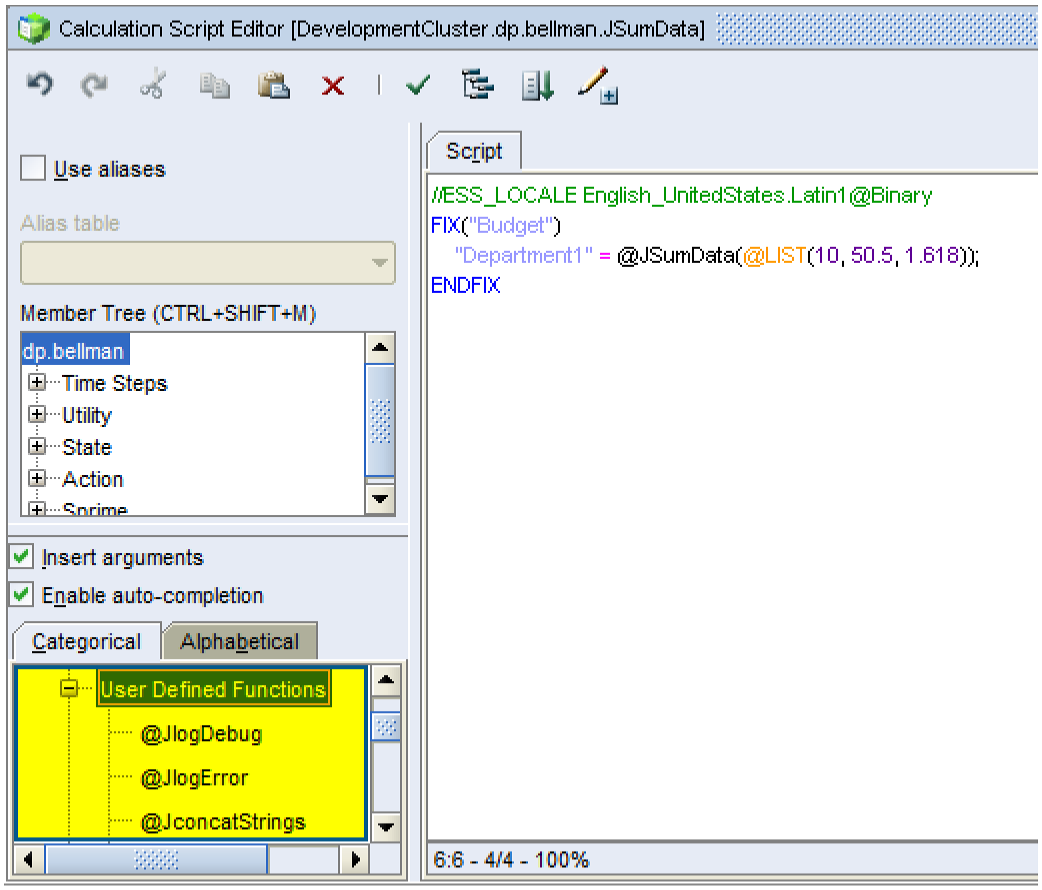
Step 3 – Essbase Calculation Scripts, Hyperion Business Rules, or Oracle EPM Calculation Manager

Once the function is registered, the new Essbase Custom Defined Functions is now ready to use in the applications. Using standard Essbase functions like @LIST to create arrays of data to be passed to java functions and @CURRMBR, @MEMBER to cast Strings back to Member references.

Code: Essbase Calculation Script

```
FIX("Budget")
        "Department1" = @JSumData(@LIST(10, 50.5, 1.618));
ENDFIX
```

The custom defined functions are available in Essbase Administration Services Console under the User Defined Function category. These are also available in the Business Rule editor and EPMA Calculation Manager.



```
//ESS_LOCALE English_UnitedStates.Latin1@Binary
FIX("Budget")
   "Department1" = @JSumData(@LIST(10, 50.5, 1.618));
ENDFIX
```