BI Publisher 11g Custom Java Extensions
Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.
Introduction ........................................................................................................................................2
About the Sample Used .......................................................................................................................3
Coding and Compiling the Java Extension ........................................................................................4
Updating the Extension Information in the Archive ..............................................................................6
Deploying the Java Extension as a WebLogic Shared Library ..........................................................9
Updating Extension Information for BI Publisher Application .......................................................16
Updating (re-deploying) BI Publisher Application ...........................................................................17
Building the Sample Report .............................................................................................................20
Conclusion ..........................................................................................................................................23
Introduction

Oracle BI Publisher can make calls to custom functions written using Java extensions. Java extensions for BI Publisher is a powerful and existing feature prior to the release of 11g version. This white paper describes the process of building, deploying and making use of Java extensions for Oracle BI Publisher 11g (11.1.1.7.0).

In order for the BI Publisher to make use of these custom extensions, the code needs to be made available in the domain library directory or deployed into WebLogic server as a shared library through Java archive files (.jar files). Once made available, calls to these custom functions can be made from within the RTF templates. Java extensions provide handy solutions when neither the data model nor BI Publisher/XSL:FO language can support the business requirements. A variety of requirements such as, complex logic evaluation, complex date computation, mathematic calculations etc. can be fulfilled through the Java extensions.

At present the custom Java extensions for BI Publisher are not supported by Oracle Support.

This white paper describes the following in detail:

- Coding and compiling a sample Java extension
- Building the Java archive (.jar file) and updating the extension related information
- Deploying the Java extension as a WebLogic shared library
- Updating the extension related information into BI Publisher application (xmlpserver.ear)
- Updating (re-deploying) BI Publisher application into WebLogic
- Building the sample report
- BI Publisher Runtime Configuration parameters to accommodate Java extensions
About the Sample Used

The sample used in this white paper makes use of Oracle Database sample HR schema. The OS used for Oracle Business Intelligence 11g (11.1.1.7.0) version is Oracle Linux 6 – 64 bit. For RTF template development, Microsoft® Word is used on Microsoft® Windows-7 desktop.

The sample report displays content from “Employees” table of HR schema along with employee grade. Grade is not part of the “Employee” table of HR schema. To determine the employee grade, Java extension is used. For this sample, employees earning less than $5,000 would be under grade “D”, employees earning between $5,000 and $10,000 would be under grade “C”, employees earning between $10,000 and $15,000 would be under grade “B” and the rest would be under grade “A”.

While calling the Java extension, the RTF template would pass the salary of the respective employee record, and the Java extension would evaluate the same and return the grade.
Coding and Compiling the Java Extension

The following sample code is used for EmpExtn.java file:

```java
// Declare the Java package here
package com.oracle.bip.extensions;

public class EmpExtn {
    public EmpExtn() {
        super();
    }

    public static final String gradeEmp(String pSal) throws Exception {
        // Declare local variables for the function
        String empGrd="Grade D";
        long empSal = 0;
        empSal = Long.parseLong(pSal);
        // Evaluate and assign the grade
        if (empSal < 5000)
            empGrd = "Grade D";
        else if (empSal >= 5000 && empSal < 10000)
            empGrd = "Grade C";
        else if (empSal >= 10000 && empSal < 15000)
            empGrd = "Grade B";
        else if (empSal >= 15000)
            empGrd = "Grade A";
        // Return the grade back to the RTF template
        return empGrd;
    }
}
```

Figure1. Code Used for Java Extension

The Java version that is supplied along with Oracle Business Intelligence 11g 11.1.1.7.0 is:

```
java version "1.6.0_35"
Java(TM) SE Runtime Environment (build 1.6.0_35-b52)
Java HotSpot(TM) 64-Bit Server VM (build 20.10-b01, mixed mode)
```

It is recommended to compile the above code through the JDK that is supplied along with Oracle Business Intelligence 11g 11.1.1.7.0 version. This code can be compiled using other versions of JDK as long as the compiled class files are compatible with the above mentioned JDK version.

The location of this JDK is `<MiddlewareHome>/Oracle_BI1/jdk`. (Example: `/home/app/obi/Oracle_BI1/jdk`). Sample command to compile the code:

```
$ /home/app/obi/Oracle_BI1/jdk/bin/javac EmpExtn.java
```
The custom Java extension should always return a string as BI Publisher cannot handle any other data types. Once the file is successfully compiled, the class file needs to be archived into a Java archive using the “jar” command. The class file should be made available in the appropriate directory as per the package declaration inside the source code.

Sample “jar” command to archive the code:

```
$ /home/app/obi/Oracle_BII/jdk/bin/jar cvf EmpExtn.jar *
```

```
oracle@sailinux EmpExtn]$ /home/app/obi/Oracle_BII/jdk/bin/jar cvf EmpExtn.jar *
   added manifest
   adding: com/(in = 0) (out= 0)(stored 0%)
   adding: com/oracle/(in = 0) (out= 0)(stored 0%)
   adding: com/oracle/bip/(in = 0) (out= 0)(stored 0%)
   adding: com/oracle/bip/extensions/(in = 0) (out= 0)(stored 0%)
   adding: com/oracle/bip/extensions/EmpExtn.class(in = 579) (out= 392)(deflated 32%)
[oracle@sailinux EmpExtn]$
```

Figure 2. Archiving the Java Extension

The simplest way to deploy the archive is to make `EmpExtn.jar` file available in the WebLogic domain library directory i.e. `<MiddleWare_Home>/Oracle_BII/user_projects/domains/bifoundation_domain/lib` (Example: `/home/app/obi/user_projects/domains/bifoundation_domain/lib`). In this method, once the file is copied the server needs to be restarted. Any changes to the archive in the future would also require the server to be restarted.

For those planning to follow the simple method, can copy the `EmpExtn.jar` as mentioned above and restart the server. Once the server is restarted, further steps can be followed from Building the Sample Report section of this white paper.

This white paper also describes the other method of deploying the archive as a WebLogic shared library in the following sections. The WebLogic documentation provided below describes in detail, about the usefulness of each of these methods:

**Oracle® Fusion Middleware Developing Applications for Oracle WebLogic Server 11g Release 1 (10.3.5) Part Number E13706-05**

Section “Adding JARs to the Domain /lib Directory” of Chapter 8 Understanding WebLogic Server Application Classloading

http://docs.oracle.com/cd/E21764_01/web.1111/e13706/classloading.htm#i1096881

Chapter 9 Creating Shared Java EE Libraries and Optional Packages

http://docs.oracle.com/cd/E21764_01/web.1111/e13706/libraries.htm#g1092115
Updating the Extension Information in the Archive

This step is required only for deploying EmpExtn.jar as a WebLogic shared library.

Notice that when the jar file is created, jar tool automatically adds the manifest related information. The manifest information is available under folder META-INF with file name as MANIFEST.MF. The next step is to edit the MANIFEST.MF file to add the extension related information.

```bash
oracle@sailinux EmpExtn]$ /home/app/obi/Oracle_BI/jdk/bin/jar cvf EmpExtn.jar *
added manifest
adding: com/(in = 0) (out= 0)(stored 0)
adding: com/oracle/(in = 0) (out= 0)(stored 0)
adding: com/oracle/bi/(in = 0) (out= 0)(stored 0)
adding: com/oracle/bi/extensions/(in = 0) (out= 0)(stored 0)
adding: com/oracle/bi/extensions/EmpExtn.class(in = 579) (out= 392)(deflated 32%)
oracle@sailinux EmpExtn]$
```

**Figure 3. Manifest Information of the Archive**

For editing the MANIFEST.MF file, 7-Zip utility is used under Windows 7 environment. Any similar tool which allows the file to be modified inside the archive could also be used.

The archive can be opened by Right clicking on the file and by choosing 7-Zip – Open archive as shown below:

**Figure 4. Manifest Information of the Archives**
This will open the archive inside 7-Zip utility. The next step is to edit the MANIFEST.MF file, by double clicking on MANIFEST folder and then right click on MANIFEST.MF and choosing Edit as shown below:

![Image of 7-Zip utility with MANIFEST.MF file being opened.]

Figure 5. Manifest Information of the Archive

Once available for editing, the following line needs to be added to the MANIFEST.MF file:

**Extension-name: EmpExtn**

The extension name could be any user defined name (in this example it is “EmpExtn”). However, the same extension name needs to be added to the xmlpserver.ear file and the same extension name needs to be used as part of the namespace inside RTF template. This is explained in detail in the later part of this white paper.
After the “Extension-name” line is added, the file needs to be saved. After exiting the editor, 7-Zip would prompt to apply the changes into the archive:

Once the “OK” button is pressed the archive is updated, it may need to be transferred to the server (as in this example the file is modified in Windows 7 desktop and needs to be transferred to the Linux server on which Oracle Business Intelligence 11.1.1.7.0 is installed)
Deploying the Java Extension as a WebLogic Shared Library

This step is required only if EmpExtn.jar is deployed as a WebLogic shared library.

For deploying the EmpExtn.jar as a shared library, this white paper used the location of this file as <MiddlewareHome>/Oracle_BI1/user_projects/domains/bifoundation_domain/lib (Example: /home/app/obi/user_projects/domains/bifoundation_domain/lib). Any other directory with appropriate permissions could be used in place of this.

The Java archive needs to be deployed as a shared library through WebLogic Admin Console. Once logged into the admin console (in this example using URL: http://192.168.199.4:7001/console), the “Lock & Edit” button needs to be clicked under “Change Center”.

Once this is done, the library needs to be deployed by clicking on Deployments.
For installing the archive file as a library, “Install” button needs to be clicked.

Figure 9. WebLogic Admin Console Deployments

Figure 10. WebLogic Admin Console Installing Shared Library
The path for the Java archive file needs to be entered inside the “Path” text field (for this example it is: 
/home/app/obi/user_projects/domains/bifoundation_domain/lib) and “Next” button needs to be clicked. Alternatively, a directory or an archive file can be selected as well, if the path is already pointing to the desired location.

Figure 11. WebLogic Shared Library Location

The archive files needs to be selected and “Next” button needs to be clicked to move forward.

Figure 12. WebLogic Shared Library Selection
The next step is to choose the deployment target. The following error message might be displayed in this screen and this message can be ignored, by clicking on “Next” button after choosing the deployment targets.

![Figure 13. WebLogic Shared Library Ignore Warning](image)

For this example, all the default values are retained in the resulting screen, by ensuring the “Name” in the “General” section is “EmpExtn”. The “Next” button needs to be clicked to move forward.
Again default values are retained in the resulting screen and “Finish” button is clicked.
Once the deployment is completed, “Activate Changes” button needs to be clicked under “Change Center” area, to complete the library deployment.
Once successfully deployed the extension would be listed as a library under deployments as shown below:

![WebLogic Shared Library Deployment Listing](image)

**Figure 17. WebLogic Shared Library Deployment Listing**
Updating Extension Information for BI Publisher Application

This step is required only if EmpExtn.jar is deployed as a WebLogic shared library. Also, this process needs to be repeated every time a new patch is applied which updates the version of xmlpserver.ear.

To make the BI Publisher application access the shared library, the extension related information needs to be entered for xmlpserver.ear file. This is similar to the way EmpExtn.jar is modified in the above steps. For this example 7-Zip utility is used to open the archive to edit the files inside.

The xmlpserver.ear file available under <Middleware_Home>/Oracle_BI1/bifoundation/jee (example: /home/app/obi/Oracle_BI1/bifoundation/jee) needs to be backed up first. This is to ensure that in case of any severe errors during deployment, we can always revert to the out of the box xmlpserver.ear.

Once the archive is opened, weblogic-application.xml which resides in META-INF directory of the xmlpserver.ear archive needs to be modified.

For weblogic-application.xml, it is better to extract the file, edit the file and then copy (drag back into the archive. Once the file is extracted, following lines need to be added, so that the library reference is made:

```xml
<library-ref>
  <library-name>EmpExtn</library-name>
</library-ref>
```

![Figure 18. Adding the Shared Library (EmpExtn) to Library Reference Listing](image-url)
Updating (re-deploying) BI Publisher Application

This step is required only if EmpExtn.jar is deployed as a WebLogic shared library.

The BI Publisher application needs to be updated through WebLogic Admin Console. Once logged into the admin console, “Lock & Edit” button needs to be clicked under “Change Center” area. After this “bipublisher” application needs to be selected by clicking on “Deployments” under “Domain Structure”. After this “Update” button needs to be clicked to re-deploy the application.

In the resulting screen, “Source path” field needs to point to:

```
<Middleware_Home>/Oracle_BI1/bifoundation/jee/xmlpserver.ear
```

(example: /home/app/obi/Oracle_BI1/bifoundation/jee/xmlpserver.ear) and Next button needs to be clicked. If the “Source path” is not correct then this can be changed by clicking on “Change Path” button.

Figure 19. Update BI Publisher Application
Figure 20. Select xmlserver.ear File

The last step is to leave the default values and click on “Finish” button.

Figure 22. Finish Updating BI Publisher Application
If there are no errors, then the deployment would complete and displays the screen similar to the following. To complete the deployment “Activate Changes” button needs to be clicked.

**Figure 23. Activate the Updated BI Publisher Application**

If all goes well the following screen would be displayed and this would complete the modification and re-deployment of “bipublisher” application (xmlpserver.ear).

**Figure 24. Activation Complete**
Building the Sample Report

As mentioned in the About the Sample Used section, for the data model Oracle Database sample schema “HR” is used. The BI Publisher data model is built using the following query:

![Figure 25. BI Publisher Data Model](image)

Once the above data model is built and sample data is saved, the next step is to build the RTF template.

Before making reference to the custom Java function in the RTF template, the XML namespace needs to be declared. In this sample the namespace is declared in the form field with the following tag:

```xml
```

Note that the extension name needs to be declared right after “namespace:” (for this sample it is “EmpExtn”). This is followed by `http://www.oracle.com/XSL/Transform/java/` and appended with package name with class name at the end. In this sample the package name is “com.oracle.bip.extensions” and the Java class name is “EmpExtn”.

Once the namespace is declared, calls can be made to the respective functions. For this sample, apart from the columns selected in the data model, an additional field called “Grade” is added. The “Grade” field is used for making the call to the “gradeEmp” function. The syntax to call the extended function is:

`namespace:function_name([Parameter1],[Parameter 2],…,[Parameter n])`

Actual tag used in this sample: `<?EmpExtn:gradeEmp(SALARY)>`
Once the template is tested and ready, the report needs to be created in the BI Publisher and the tested template needs to be uploaded to the repository. Before executing the report, the runtime property of **Disable External References** needs to be set to false. Setting “Disable External References” to “False” allows sub-templates and Java extensions to be called through the templates.

The first step is to click on Properties as shown below:

![Figure 27. Report Properties](image)

**Figure 27. Report Properties**

The next step is to click on “Formatting” tab and change the value to “False” (for “Disable External References”) under “FO Processing” section as shown below:
Figure 28. Runtime Formatting Properties

At this time the report can be executed and if everything is done correctly, the result would be something similar to the following:

Figure 29. Sample Report Output
Conclusion

By utilizing the powerful feature of Java extensions for BI Publisher, the power of Java programming language can be added to the BI Publisher customizations. The centralized libraries built and deployed into WebLogic, could be used across templates and reports. This would ease the maintenance of complex business logic and increase the productivity.

Oracle consulting has had many experiences in implementing custom Java extensions for BI Publisher. If you are interested in Oracle consulting to discuss more in detail about the implementation and review of your reporting or customization needs, please contact Shankar Duvvuri (shankar.duvvuri@oracle.com), Senior Principal Consultant, Oracle Business Intelligence group.