WSO2 Application Server 5.1.0 – Classloading guide [Draft]

1. Introduction

WSO2 Application Server 5.1.0 provides a new concept called “Runtime Environment” to control classloading per server/application. Users can define own runtimes for advanced usages but default runtimes are sufficient for most of the cases. There are 3 default runtimes on AS.

1. **Tomcat Environment** – This is the minimal runtime and identical to pure Tomcat runtime. Only Tomcat, Servelet, JSP, EL, JSTL are available on server level Classpath. If a user wants additional Jars they need to be packaged with web application or should be placed on Tomcat Environment extension directory.

2. **Carbon Environment** - Tomcat Environment + Carbon runtime. This does not provide CXF or Spring dependencies. If a user wants additional Jars they need to be packaged with web application or should be placed on Carbon Environment extension directory.

3. **CXF Environment** - Tomcat Environment + CXF + Spring. This does not provide top access Carbon runtime. If a user wants additional Jars they need to be packaged with web application or should be placed on CXF Environment lib directory.

2. Configuration

There is a new configuration file called webapp-classloading introduced to configure Classloading behavior per application/server basis. This file need to be placed on META-INF directory of a web application.

e.g – META-INF/webapp-classloading.xml

If there is no configuration file provided application will deploy on default “Carbon” environment. It's mandatory to use webapp-classloading.xml file with correct configuration to use other runtime environments except “Carbon”.

This configuration file takes following format.

<Classloading xmlns="http://wso2.org/projects/as/classloading">
<Environments>{Runtime Environment Names} ***</Environments>***
</Classloading>

e.g – To specify CXF as the runtime environment users should have the following configuration.

```
<Classloading xmlns="http://wso2.org/projects/as/classloading">
   <Environments>CXF</Environments>
</Classloading>
```

e.g – To specify CXF and Carbon as the runtime environments users should have the following configuration.

```
<Classloading xmlns="http://wso2.org/projects/as/classloading">
   <Environments>CXF,Carbon</Environments>
</Classloading>
```

Notes :-

- It is possible to specify list of environments to achieve some functionalities.
  - E.g – To access some Carbon features on a CXF applications it is possible to specify environment as “CXF,Carbon”

3. Runtime Environment extensions

It is a very common requirements to share dependencies among number of applications without packaging with each and every application. In such cases following extensions directories can be used to place common dependencies.

1. Tomcat Environment - **${carbon.home}/lib/runtimes/ext/**
2. Carbon Environment - **${carbon.home}/repository/components/lib**
3. CXF or any Custom Environment – Environment's lib directory.
   1. e.g - ${**carbon.home}/lib/runtimes/cxf**
   2. e.g - ${carbon.home}/lib/runtimes/ABC

Note – Dependencies placed on Tomcat Environment extension directory are visible to all the environments and it is required to identify the impact before placing dependencies on above environments. As an example placing incompatible Spring Dependencies on ${carbon.home}/lib/runtimes/ext/ can cause issues on CXF runtime environment (CXF environment also contains Spring dependencies). In such cases those incompatible dependencies should be packaged inside the web application and should deploy with Tomcat Environment configuration, then other application won't effect.

4. Adding custom Runtime Environments.

By adding new <ExclusiveEnvironments> entry to the {carbon.home}/repository/conf/tomcat/webapp-classloading-environments.xml file users can define custom runtimes. But this is recommended for advanced usages only and users should aware about the impact of these modifications.

e.g – Define a custom Runtime Environment for Spring

1. Modify {carbon.home}/repository/conf/tomcat/webapp-classloading-environments.xml file with following entry.

```
<ExclusiveEnvironments>
    <ExclusiveEnvironment>
      <Name>Spring</Name>
      <Classpath>${carbon.home}/lib/runtimes/spring/*.jar;$
                          {carbon.home}/lib/runtimes/spring/</Classpath>
    </ExclusiveEnvironment>
  </ExclusiveEnvironments>
```

2. Create and copy related Spring dependencies to the {carbon.home}/lib/runtimes/spring/ directory.

3. Now it is possible to use above "Spring" Runtime Environment per application basis by adding **<Environments>CXF,Carbon</Environments>** entry to the **META-INF/webapp-classloading.xml**

file.

Note – As mentioned above users are advised to study all impacts before introducing custom Runtime Environments to the system. Please refer following example.

e.g – Assume a AS instance contains following configuration.

- CXF (provide runtime by AS) - Contains CXF 2.7.3 and Spring 3.0.7 dependencies.
- Spring (user defined run time) - Contains Spring 3.2.1 dependencies.

Assume an application having following configuration
<Environments>CXF,Spring</Environments>
Above configuration can be result into one of the following issues.

1. Application Classpath contains dependencies from two Spring versions as 3.0.7 and 3.2.0, this is not a recommended approach by Spring project and will cause number of Classloading issues.

2. CXF 2.7.3 itself use Spring 3.0.7, there is a possibility that particular CXF version not

tested/compliant with Spring 3.2.0 and will cause for various issues. Here it's required to consider about CXF project recommendations to find compatible Spring version.

5. Miscellaneous

- AS 5.1.0 ships core CXF 2.7.3 dependencies and core Spring 3.0.7 dependencies under CXF runtime environment. If users want to upgrade to different CXF versions they are two options available.
    - Upgrade CXF runtime environment – Users can remove existing CXF/Spring and dependent Jars from "{carbon.home}/lib/runtimes/cxf" directory and can place new CXF/Spring Jars.
    - Instead of upgrading server level CXF dependencies, users can package all the required Jars with in the Web-INF/lib directory of the application and deploy with "Tomcat" runtime environment.
- If user want to add optional CXF/Spring jars, they can copy them into "{carbon.home}/lib/runtimes/cxf" directory but need to ensure those new dependencies are compatible with existing CXF/Spring Jars.