# J2EE - Confirm analysis configuration

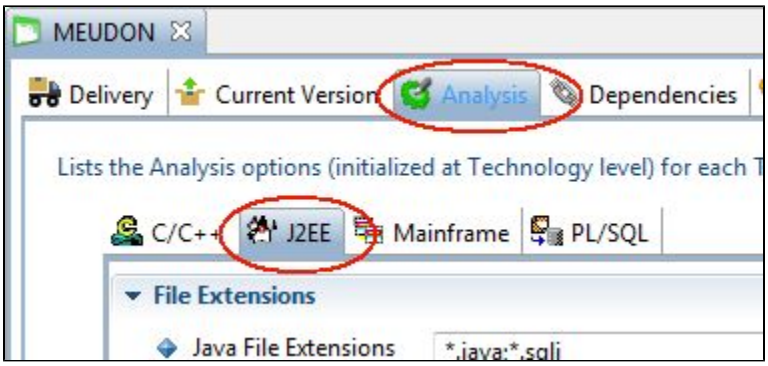
**On this page:**



## General settings

### Default source code options

Using the **Technology level** or **Application level** options, validate the **Analysis settings** for J2EE packages. Make any update as required. These settings apply to the Technology or Application as a whole (i.e. all Analysis Units). There is no override at the Analysis Unit level.




# Analysis Unit configuration

The DMT extracts relevant information used to create the automated analysis configuration from the Java project file. Currently the CAST DMT supports these build project files:

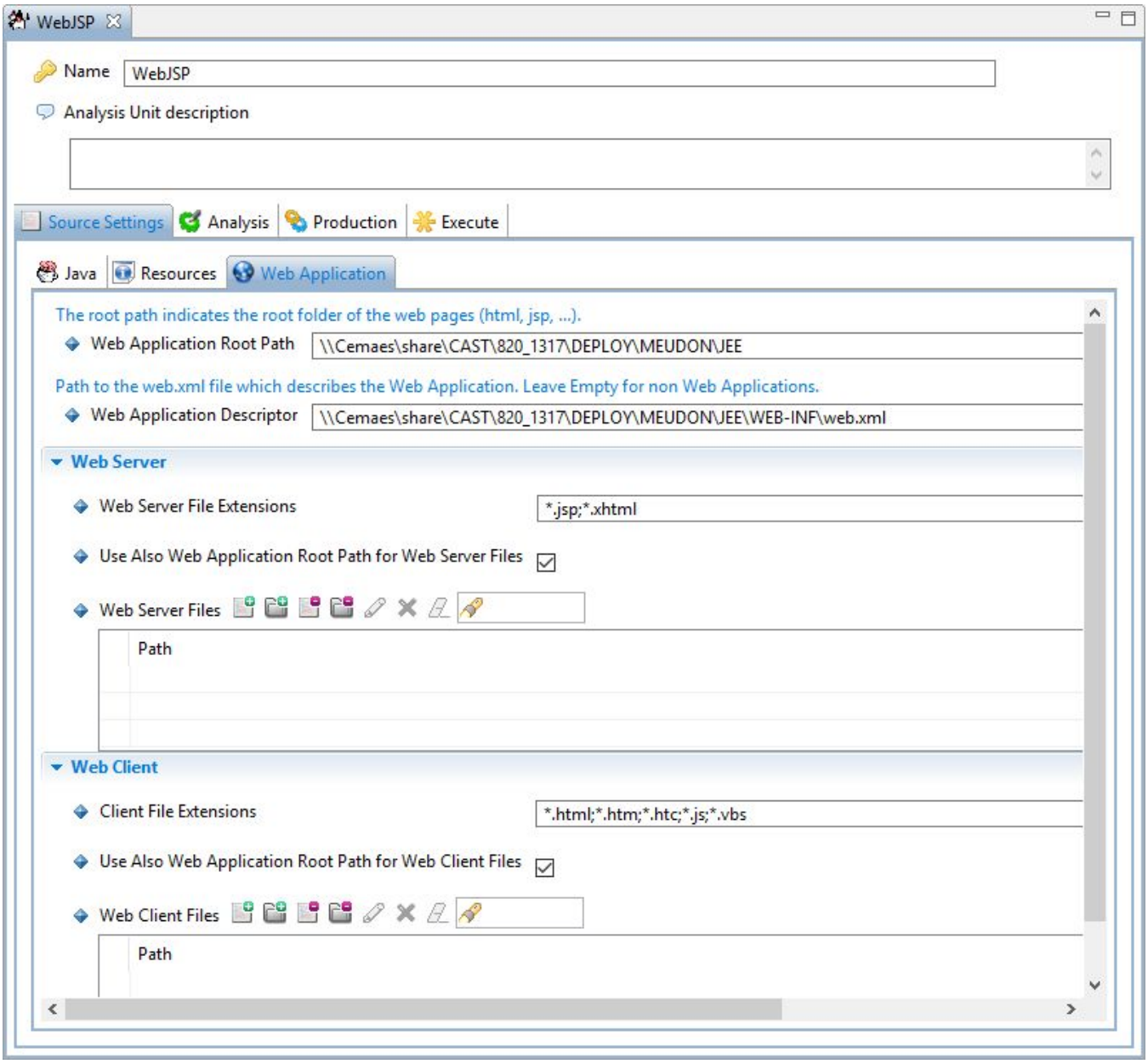
- Eclipse - all version greater than 3
- Maven - version 2 and 3

For any other build project format (e.g. Apache -ant) CAST AIP will not be able to automatically retrieve build information: no Analysis Unit will be created and no analysis configuration provided. To address this situation, the Analysis Unit and the analysis configuration should be created manually. This case falls outside of the out-of-the-box support and is out of scope of a standard analysis.

To inspect the auto-generated analysis configuration, you should review the settings in each Analysis Unit:

## Source Settings

This tab and its child tabs (Java, Resources and Web Application) show the location of each type of source code in the JEE Analysis Unit - this is determined automatically by the CAST Delivery Manager Tool. You should, however, review the configuration and make any changes you need:

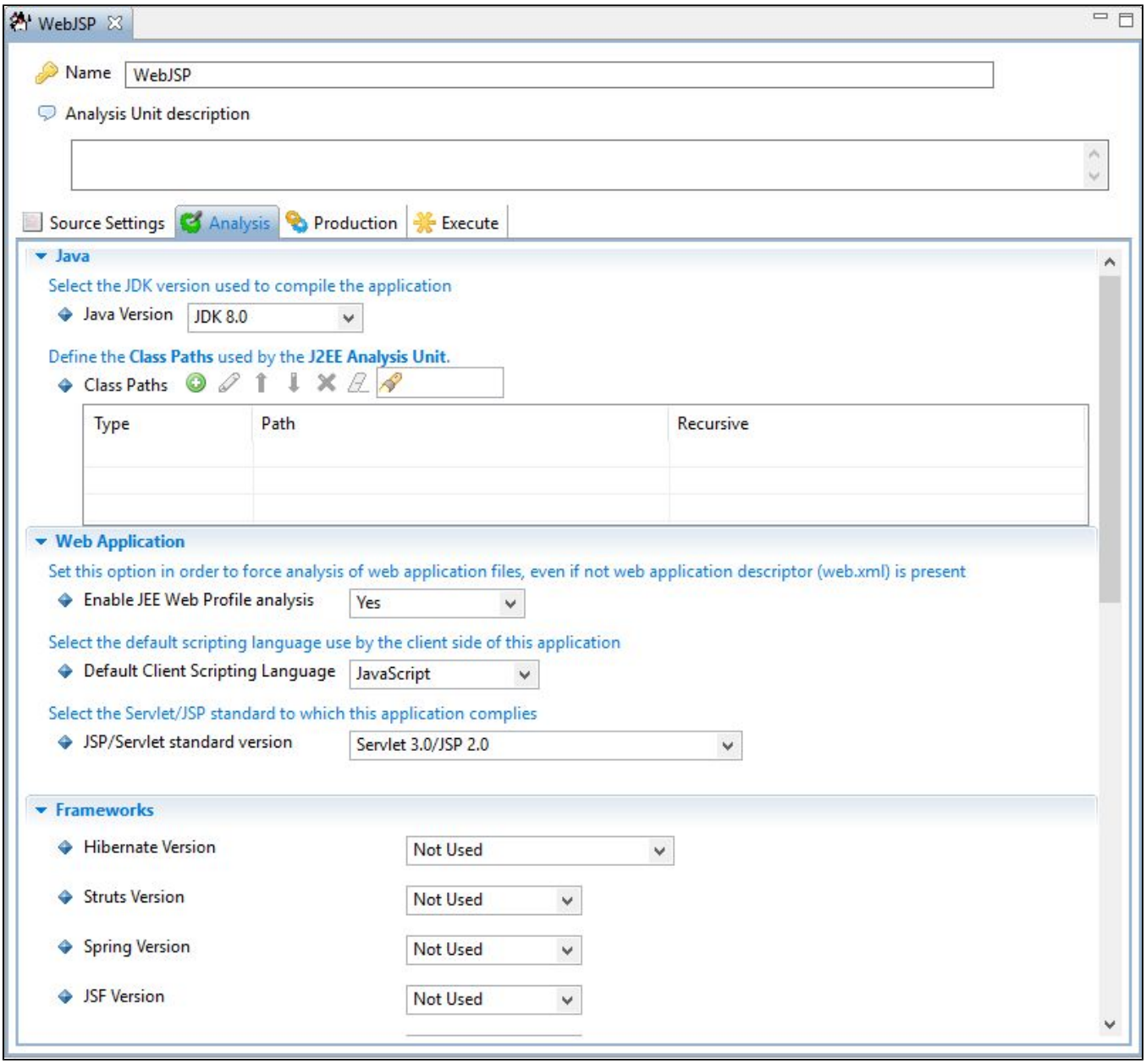



Note about the **Web Application Root Path**

CAST AIP supports the analysis of web applications that **do not** contain a **web.xml** application descriptor. In this situation, the "Web Application Root Path" (i.e. the location of web content files such as *.HTML and *.JSP) can be automatically discovered by the analyzer based on the location of the web files on disk. To this end, the field **Web Application Root Path** at **Analysis Unit level** is now an optional field in the CAST Management Studio. If the source code does not contain a web.xml file, the Web Application Root Path is now automatically discovered by exploring each Analysis Unit's **project** path (i.e. the location of the **.project** file) on disk, however, it can be overridden manually.

## Analysis

The settings in this tab govern how the source code is handled by the CAST analyzer:




## Java

### Java Version

The JDK version used to compile the application is automatically discovered by the DMT through inspection of the application project files. If the project files are missing the information must be derived from the inspection of the source code or by inquiring with the Application Team.

### Class Paths

Class Paths are automatically discovered by the DMT from the inspection of the project files. When the project file is missing, Class Paths must be added manually to the Analysis Unit and must be discovered via inspection of the delivered source code.

## Web Application

### Enable JEE Web Profile Analysis

By default this option will be set to YES:

| | |
|---|---|
| YES | When set to the default **YES** position, the analyzer is capable of analyzing web files (JSP, XHTML, HTML, JS etc.) even if no **Web Application Descriptor (web.xml)** is present in each Analysis Unit in the Application. These web files are instead identified for analysis by exploring each Analysis Unit's Analysis Unit's **project** path (i.e. the location of the **.project** file). |
| NO | When set to the **NO** position, the behaviour will revert to pre CAST AIP 8.2.x. In other words a **Web Application Descriptor (web.xml)** must be present in the Application and defined in the relevant field in each Analysis Unit in order that web files (JSP, XHTML, HTML, JS etc.) are analyzed. |

### Default Client Scripting Language / JSP/Servlet Standard version

Even when the build project format is supported, the CAST AI Admin must configure the "Default Client Scripting Language" and the "JSP/Servlet Standard version". This information can be acquired during the Application qualification step by asking the app team or through source code inspection if this information is missing.

Alternatively you can discover the correct setting as indicated below:

**Default client scripting language:** In 90-95% of J2EE application the client scripting language is JavaScript. To validate this you should find a JSP or HTML file with a SCRIPT tag that does not have a language definition (do a grep search for "<SCRIPT>") and analyze the language to determine if it is JavaScript or Visual Basic. For an application that mixes both languages, you will need to find a file with both SCRIPT tags that do not have a language specified and SCRIPT tags that do have a language specified. You then need to select the language not specified in the SCRIPT tag as the default language.

**JSP/Servlet version:** to find this information, open the **web.xml** file and look at the DTD version used to define the Servlet, for example:

```
<!DOCTYPE web-app PUBLIC
   "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
```

Using this information, see the table below for equivalence:

| Servlet version | JSP version |
|---|---|
| 2.1 or 2.2 | 1.1 |
| 2.3 | 1.2 |
| 2.4 | 2.0 |

## Frameworks

For more information about frameworks see J2EE Framework analysis.

Discovery of Hibernate, Struts and Spring relies on the scanning of the **.classpath files (Eclipse)** and the **pom.xml files (Maven)**.

When out-of-the-box supported frameworks are present but the version of the framework does not match one of CAST supported ones, the proposed configuration recommends the best possible matching profile as a workaround. Impact should be expected on the Transaction count. Improving the configuration to address such gap require ad hoc investigation. The following associations are automatically proposed when a supported framework type cannot be discovered with certainty:

- **Hibernate**
  - presence of hibernate.jar → version 2.x
  - presence of of hibernate3.jar → version 3.6
  - presence of hibernate4.jar → version 4.x
  - no match → No Value*
- **struts**
  - presence of struts1.1.jar, struts.1.2.jar, struts.1.3.jar → version 1.1
  - presence of struts2.jar → version 2.x
  - no match → No Value*****
- **Spring**
  - presence of spring.jar and spring-core.jar → 1.x
  - presence of only spring.jar → 2.x
  - presence of org.springframework-core.jar → Not Used
  - - no match → No Value*****

***When no value is set the actual analysis configuration setting is inherited from the Application level settings**

Limitations exist when mixing frameworks.

The discovering of frameworks, other than Hibernate, Struts or Spring, supported out-of the-box by CAST AIP, is based on pattern matching on the .jar files delivered. Configuration for those is a simple **Yes** (if a .jar file matching is found) and **No** otherwise. When unsupported frameworks (i.e. not supported out-of the box) are part of the delivery, a Custom Environment Profile need to be created to enable correct processing. See 2.2.2. Links and Transactions fo r further details about how to address missing links situations.

To identify unknown frameworks or custom frameworks, there are various methods:

- Find all the XML files and look at the DTD they are referring to. This allows you to find frameworks like Spring, Hibernate... (You can use UltraEdit or a grep command to find the pattern DTD inside XML files in one go). Examples:

| Framework | DTD |
|---|---|
| Hibernate | <!DOCTYPE hibernate-mapping PUBLIC<br>"-//Hibernate/Hibernate Mapping DTD 2.0//EN"<br>"http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd"><br><br><!DOCTYPE hibernate-configuration PUBLIC<br>"-//Hibernate/Hibernate Configuration DTD 2.0//EN"<br>"http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd"> |
| iBATIS | <!DOCTYPE daoConfigPUBLIC "-//iBATIS.com//DTD DAO Configuration 2.0//EN""http://www.ibatis.com/dtd/dao-2.dtd"> |
| Spring | <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd"> |

- Look at the libraries: the names of the libraries can give you an indication.Examples:

| Framework | Libraries |
|---|---|
| Hibernate | hibernate3.jar |
| iBATIS | ibatis-common-2.jar<br>ibatis-dao-2.jar<br>ibatis-sqlmap-2.jar |
| Spring | spring-1.2.7.jar |

- Look at the different extensions used throughout the entire application. This information will also allow you to verify if the default extension list used by the analyzer to parse client and server source code is complete.Examples:
  - .jhtml: for Java HTML (See http://en.wikipedia.org/wiki/JHTML). This is used with Dynamo, a Java-based Web application server
  - .jspx: JSPX pages are a special type of JSP page: They are pre-processed by the JSPXServlet, and converted into regular JSP pages (See: http://stackoverflow.com/questions/28235/should-i-be-doing-jspx-instead-of-jsp)
  - .jspf: for JSP Page Fragment
  - .jrxml: for Jasper Report (http://en.wikipedia.org/wiki/JasperReports#JRXML)

If you have detected this framework through an XML file, you can start to customize the support of this XML (see chapter 9.1 Manage XML files), but in any case, we recommend that you read up on how it works and how it is configured.

## WebServices (WBS) and Enterprise Java Bean (EJB)

Auto discovery and configuration of Web Services (WBS Services) and Enterprise Java Bean (EJB) is not currently supported. The CAST AI Admin will have to gather the required information from the application team or inspect the delivered source code to determine the proper configuration of these parameters.

To find out if your application implements web services, simply search for files with the extension .wsdl or .wsdd in the source file directories. Usually, these files are located in the META-INF directory, but as the source folders do not always match the production folder tree, we recommend searching all the source files.

### Custom Web Service Environment Profile

This step is required only if you have a web service in your application.

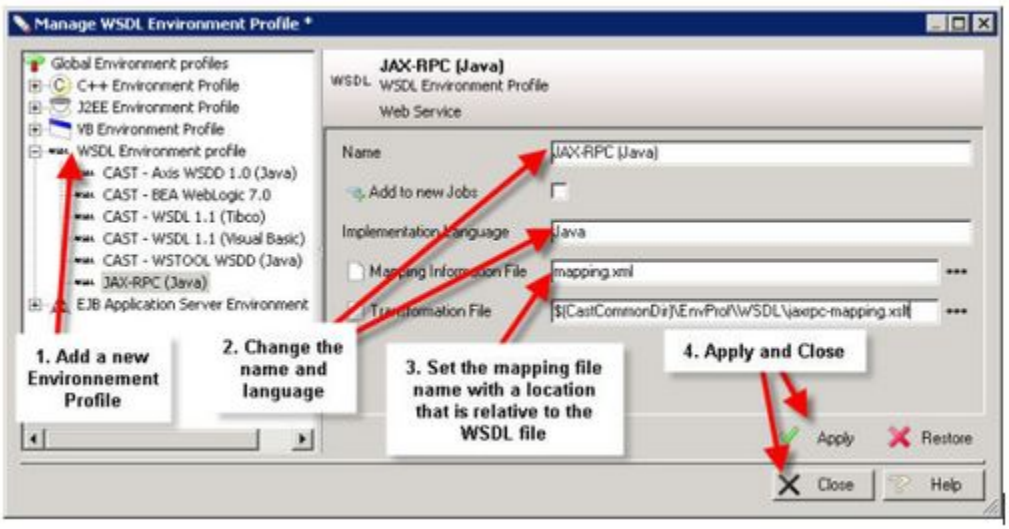
First, do not forget to select the wsdl file(s) in the Source Settings > XML files of the Analysis Unit.

In a J2EE application, the implementation language is Java. Different profiles have been defined according to the most common implementations. Each of them are related to a specific Mapping Format File that defines the relationship between the web services defined in the WSDL and the Java classes and method that are exposed through this Web Service. In the Java world, you could come up against the following frameworks that are used to expose Web Services:

- Apache Axis (de facto standard in the Java World): this implementation uses .wsdd extensions for the mapping file. Note that the WSDD file is not a substitution for the WSDL file that is a language independent file used to describe a web service so that it can be used by any type of client. But if you have a WSDD file it does mean that you have an Axis implementation.
- Frameworks based on JAX-RPC that provide the JAX-RPC mapping file. These frameworks rely on a WSDL file to describe the web service and a JAX-RPC mapping file: an XML file that refers to j2ee_jaxrpc_mapping_1_1.xsd:

```
<?xml version="1.0"encoding="UTF-8"?>
   <java-wsdl-mapping xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"version="1.1"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
         http://www.ibm.com/webservices/xsd/j2ee_jaxrpc_mapping_1_1.xsd">
```

  - To handle it, add a new Web Service environment profile with the name "JAX-RPC (Java)", select the JAX-RPC deployment descriptor file with a path relative to the directory that contains the WSDL as the Mapping Information Files and select the file <CAST Common folder>\EnvProf\WSDL\jaxrpc-mapping.xslt as the Transformation File.In the following screenshot, we assume that the mapping file (mapping.xml) is in the same directory as the WSDL.




- For other implementations you must find the mapping file: an XML file that contains the service and operation and also the fully qualified Java class and methods that implement this service. Usually they are located in the same directory as the WSDL. To customize a profile, refer to Customize an XSLT file for Web Services.

## Custom EJB Framework

> This step is required only if you have EJBs (session, message driven and/or entity) **version 2.0 or 2.1** in your application, in which case you will need to select the EJB 2 Environment Profile.

First, do not forget to select the ejb-jar.xml file(s) in the Source Settings > XML files of the Analysis Unit

Then, you need to select the application server that is being used. Note that **when no EJB entities** are used, you can select the option "**EJB 2 - No Entities**", there is no need to select an application server since all information about EJB Session and Message Driven Bean are contained in the ejb-jar.xml.

First check if your ejb-jar.xml contains entities or not:

```
<ejb-jar>
        <enterprise-beans> ...
                <entity> // ENTITIES ARE PRESENT ...
```

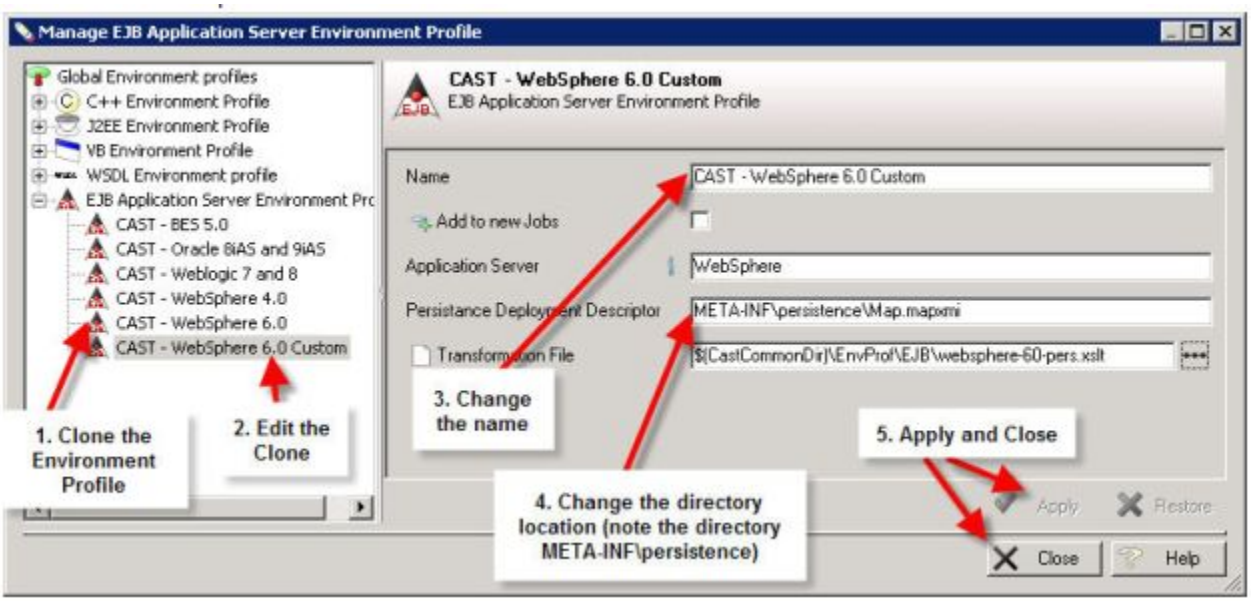
Then you must identify the application server and check that it is already supported or create a specific Environment Profile.
The best way is to get this information from the customer. If this is not possible, here are some characteristics of each Application Server that may help you to find the information you require:

**NOTE**: usually, the deployment Mapping File is located in the META-INF directory. However, this is not 100% certain, so make sure you search the entire application folder.

| ***Application Server*** | ***Typical Deployment Mapping File Name*** | ***DTD in the Deployment Map File (bold text indicate information that can help you to find the Application Server and version)*** |
|---|---|---|
| Borland | ejb-borland.xml | <!DOCTYPE ejb-jar PUBLIC "-//Borland Software Corporation//DTD Enterprise JavaBeans 2.0//EN" "http://www.borland.com/devsupport/appserver/dtds/ejb-jar**_2_0-borland**.dtd"> |
| WebSphere 4.0 | Map.mapxmi | No DTD! |
| WebSphere 6.0 | Map.mapxmi | No DTD! |
| Oracle AS 8iAS & 9iAS | oracle-ejbxxx-.xml | <!DOCTYPE oracle-descriptor PUBLIC "-//Oracle Corporation.//DTD Oracle **1.1**//EN" "oracle-ejb-jar.dtd"> |
| Weblogic 6.x | weblogic-ejbxxx.xml | <!DOCTYPE weblogic-ejb-jar PUBLIC "-//BEA Systems, Inc.//DTD **WebLogic 6.0.0** EJB//EN" "http://www.bea.com/servers/**wls600**/dtd/weblogic-ejb-jar.dtd"> |
| Weblogic 7 & 8 | weblogic-ejb-jar.xml | <!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD **WebLogic 7.0.0** EJB//EN' 'http://www.bea.com/servers/**wls700**/dtd/weblogic700-ejb-jar.dtd'> |

**IMPORTANT**: The analysis expects the Deployment Mapping File to be in the META-INF directory. So what can you do when this is not the case? Since the default persistence configuration (the lines Application Server - Persistent Label that appear in this menu) can't be changed, you need to create another one and set the correct path.

- For example: if the Map.mapxmi file descriptor of a WebSphere Application Server is located in the directory META-INF/persistence instead of META-INF, you need to do the following: Open the Environment Profiles through Infrastructure View > Knowledge Base > Environment Profile > Manage
- Clone the Web Sphere Environment Profile




- And change the directory
- Then select the EJB Version 2.x - Custom Environment Profile and select this new Environment Profile though Custom EJB 2 Environment Profile.

**Specific required files**

Depending on the Application Server, you may need to have other files that depend on the Deployment Mapping File listed in the previous table to make the analysis function correctly. If they are not present, you must request them from the customer:

- WebSphere:
  - Schema.dbxmi
  - <table name>.tblxmi
- Weblogic:
  - weblogic-rdbms-jar file type: usually weblogic-cmp20-rdbms-jar.xml
- toplink-ejb-jar file type: usually toplink-ejb-jar.xml that can reference XML files like: <ejb_name>.xml

**Other Application Servers or versions:**

- *Resin Application Server*: the ejbresin-jar.xml configuration file used by Resin is similar to the ejb-jar.xml file but does not entirely meet the JSR 26 specification. This file begins with the tag <resin-ejb> and not the <ejb-jar> tag as specified in the JSR 26. To manage this case, you need to patch this file: change all occurrences of the tag <resin-ejb> to <ejb-jar> and then reference it as you do for the ejb-jar.xml

If the Application Server does not figure in the list provided in the Options page: refer to section § Customize an XSLT file for EJB.

In the case you have entities, do not forget to add a dependency: Source = this analysis unit, Target = database.

# Advanced J2EE Configuration

Please see the following pages for more information about advanced J2EE analysis configuration:

- Customize an XSLT file for EJB
- Customize an XSLT file for Web Services
- Customize the cast-tag.extensions.xml
- Manage annotations
- Manage XML files

# Manual configuration

When the automated configuration fail or it is not possible (i.e. project build file format not supported), the CAST Admin must resort to an in-depth inspection of the source code to build the Analysis Unit and configure them manually.  For typical J2EE application this would require identify all application components including:

- A presentation layer
- A persistent/session layer
- A business layer
- (Optionally) a web service layer
- (Optionally) other components