# JEE File Discoverer 2.0

**On this page:**



**Target audience:**

CAST Administrators


**Summary**: Information and Release Notes about the **JEE File Discoverer** for the CAST Delivery Manager Tool. This extension is a **file system discoverer**.


# Extension description

This discoverer detects projects based on the presence of java files.

## In what situation should you install this extension?

This extension should be used when the java project files such as .project or pom.xml are not delivered with the source code (without these files, the discoverers provided "out of the box" in the CAST Delivery Manager Tool cannot detect all java files). In summary the extension provides a "catch-all" to ensure that all java files will be packaged for analysis

When the extension is installed, you can choose to activate it or not in each package you create in the CAST Delivery Manager Tool. To do so, click the **Package Configuration tab** and define the type of projects to discover in the package (refer to the **Extension interface** paragraph below).

Please be aware that this extension is only intended to be used as a fall-back for projects where "out of the box" discoverers provided in CAST AIP are not enough. It should be used **in parallel with** or **instead of** the other discoverers. On large projects it can produce sub-optimal analysis configurations, with some side effects that can impact the stability of results.

## Technical information

This extension is designed to detect java files in the following scenarios:

### Case 1

The package identified in the first java file matches the folder hierarchy. In this situation **one project** is created by the discoverer as follows:

- the project name will be the name of the parent folder of the folder containing the main package (i.e. com)
- the root path of the project will be the same as the project name
- the folder containing the main package (i.e. com) will be selected as the source code

For example, for a java file with the following folder hierarchy **C:\temp\project1\src\com\cast\test.java**, the project will be named "**project1**", the root path will be "**project1**" and the selected folder is "**project1/src**".

### Case 2

Similar to **Case 1** above, when a sub-folder "src" and a sub-folder "test" are located inside the same folder "**project1**", the project created by the discoverer ("**project1**") will reference the two folders for the source code: "**project1/src**" and "**project1/test**".

### Case 3

In this situation the java files do not have a package and one project will be created by the discoverer for **each java file that is discovered** (and not each folder). The project name uses the same name as the discovered java file.

> This is known as an **Unnamed Package** and will be recorded as such in the CAST Delivery Manager Tool log - see Unnamed Package below for more information.

### Case 4

In this situation, the package doesn't match the folder hierarchy (for example, **a.b.c.class1** but the folders **a** and **b** are missing). A warning is displayed and the folder **c** is considered as the main package. The **parent folder** of **c** is selected for the source code and the **grand-parent** is used for the **project name** and **root**.

> This is known as an **Incomplete Package** and will be recorded as such in the CAST Delivery Manager Tool log - see Incomplete Package below for more information.

### Case 5

Similar to **Case 4** above, however, the java file is not located in the correct folder. In this situation (known as an "UnmatchingPackage") the behavior of the discovery is the **same as Case 4**, however, the log file generated by the CAST Delivery Manager Tool will contain the following to indicate the nature of the discovery:

> This is known as an **Unmatching Package** and will be recorded as such in the CAST Delivery Manager Tool log - see Unmatching Package below for more information.

### Case 6

In this situation, each Java branch is assigned to a separate project, for example:

- c:\temp\Project1\src\com\cast\file1.java
- c:\temp\Project1\x\y\z\src\com\component\file2.java

In the above situation (known as a ParentProject), two projects will be created.

> This is known as a **Parent Project** and will be recorded as such in the CAST Delivery Manager Tool log - see Parent Project below for more information.

## About dependencies

No **dependencies** (in the CAST Management Studio > Application editor > Dependencies tab) will be automatically created based on the use of this extension since the "discovery" is based on the presence of java files, rather than Maven or Eclipse projects. As such, you may need to manually create dependencies between Analysis Units to achieve any inter-Analysis Unit links you require.

## CAST AIP Upgrade

When upgrading from **CAST AIP 7.0.x** to a **compatible version** (see section below), the java projects already discovered in **CAST AIP 7.0.x** and all previous configuration changes made in the CAST Management Studio will be preserved.

In order to respect this requirement and the project renaming conventions from v7.0.xx, the extension will remove any character from the project name that is not included in the following character set:

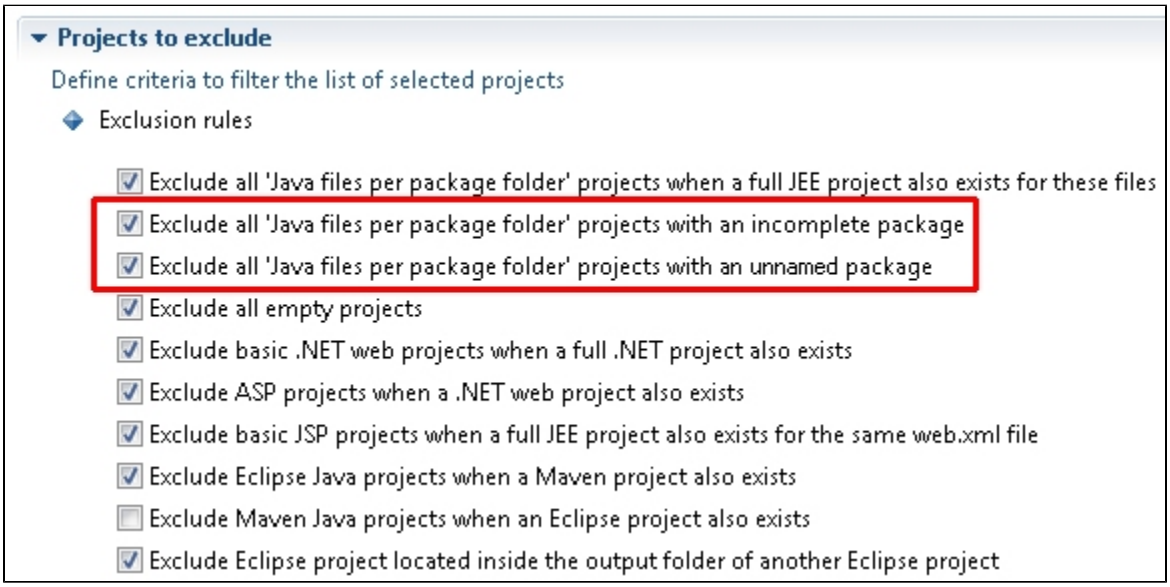
```
01234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.!:/ù^é~#´{[|è_çà@]=}+°$£*µ%§ñõìùòîôûê€
```

# What's new?

## 2.0.0

### New features

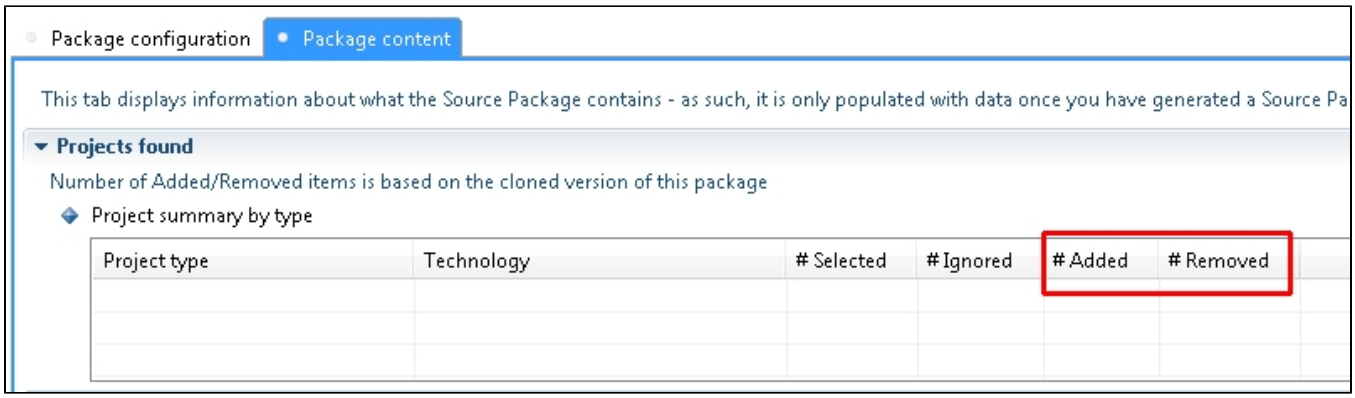
- The JEE File discoverer has been completely rewritten for release 2.0.0 to improve the accuracy of the projects it discovers. Please read the Technical Information section to understand how the discoverer functions and identifies projects.
- Project discovery should now be more accurate.
- Two new additional **exclusion rules** have been added to the CAST Delivery Tool Manager interface - see below for more information. Note that when upgrading from **JEE File Extension 1.x** to **JEE File Extension 2.0**, these options are not automatically ticked.



### Upgrading from JEE File Extension 1.x

If you have previously been actively using the **JEE File Extension 1.x** with existing Applications and you decide to install **JEE File Extension 2.x** and use it with these Applications, you will find that a new packaging action on the same source code may produce different packaging results, i.e the list of projects discovered may change.

The CAST Delivery Manager Tool will show you which projects have been **Added** and **Removed** in comparison to the previous packaging action in the **Package Content tab**:




# Bug fixing

Please see JEE File Discoverer 2.0 - Bug Fix List for more information about bugs that have been fixed in this release.

# CAST AIP compatibility

This extension is compatible with:

| CAST AIP release | Supported |
|---|---|
| 8.2.x | ✓ |
| 8.1.x | ✗ |
| 8.0.x | ✗ |
| 7.3.x | ✗ |

# Download and installation instructions

Please see:

- http://doc.castsoftware.com/display/EXTEND/Download+an+extension
- http://doc.castsoftware.com/display/EXTEND/Install+an+extension



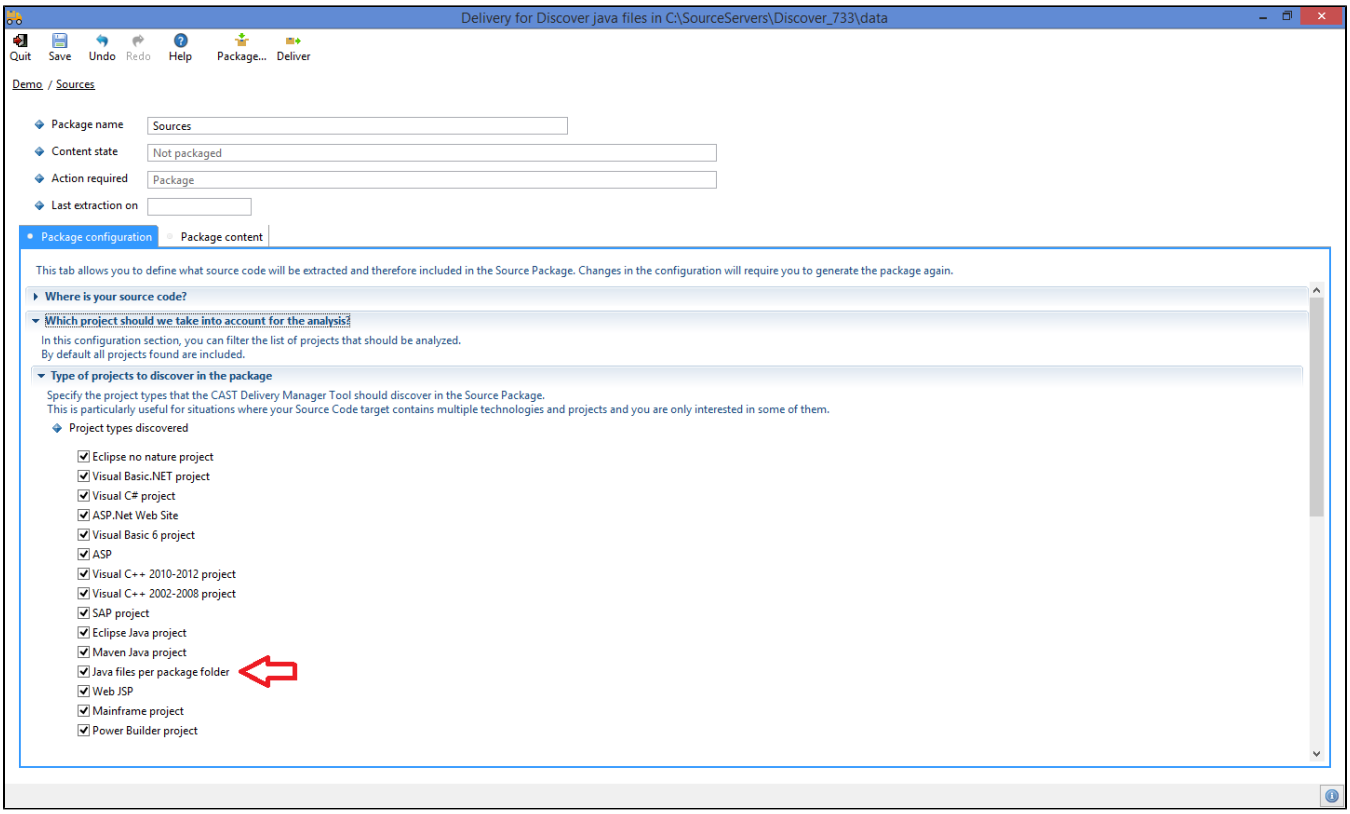
- This extension contains a **File discoverer** and you should take note of the specific instructions in the installation guide that explains how to package your source code with the CAST Delivery Manager Tool when you have an **existing Version**.
- The latest release status of this extension can be seen when downloading it from the CAST Extend server.

# Extension interface

The following screen shots show the differences in the product when the extension is installed:

- in the CAST Delivery Manager Tool, in the **Package configuration tab**, the interface contains a new item in the list of "**Project types discovered** ":

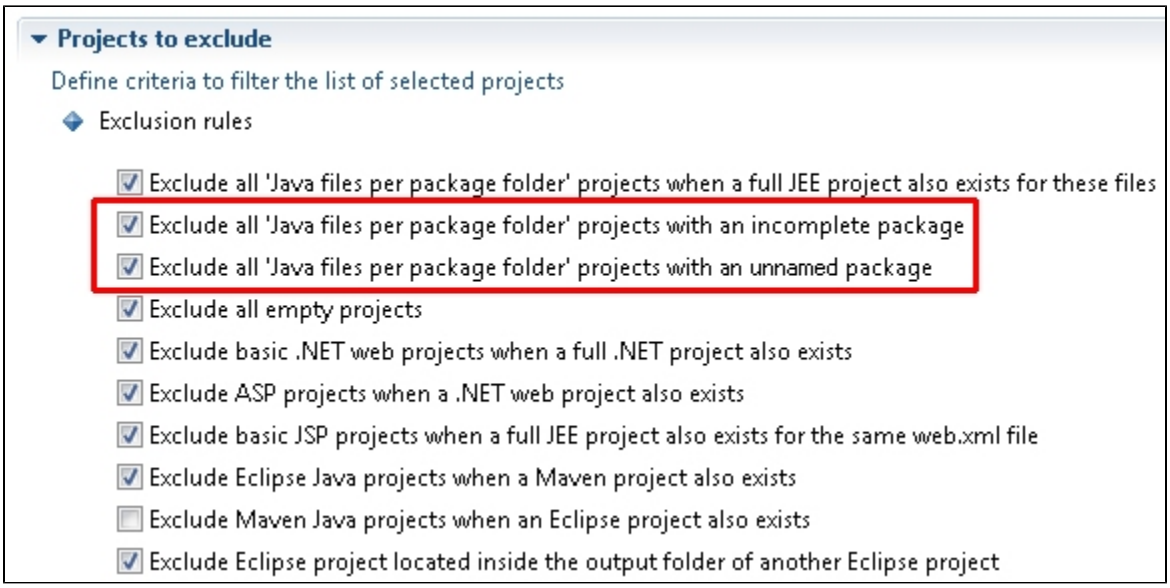
*Click to enlarge:*




When the extension is installed after the creation of the package, the item shown above is not selected by default. If you activate it after the Package action has been executed, you must re-package (without forcing the re-extraction) to discover the projects.

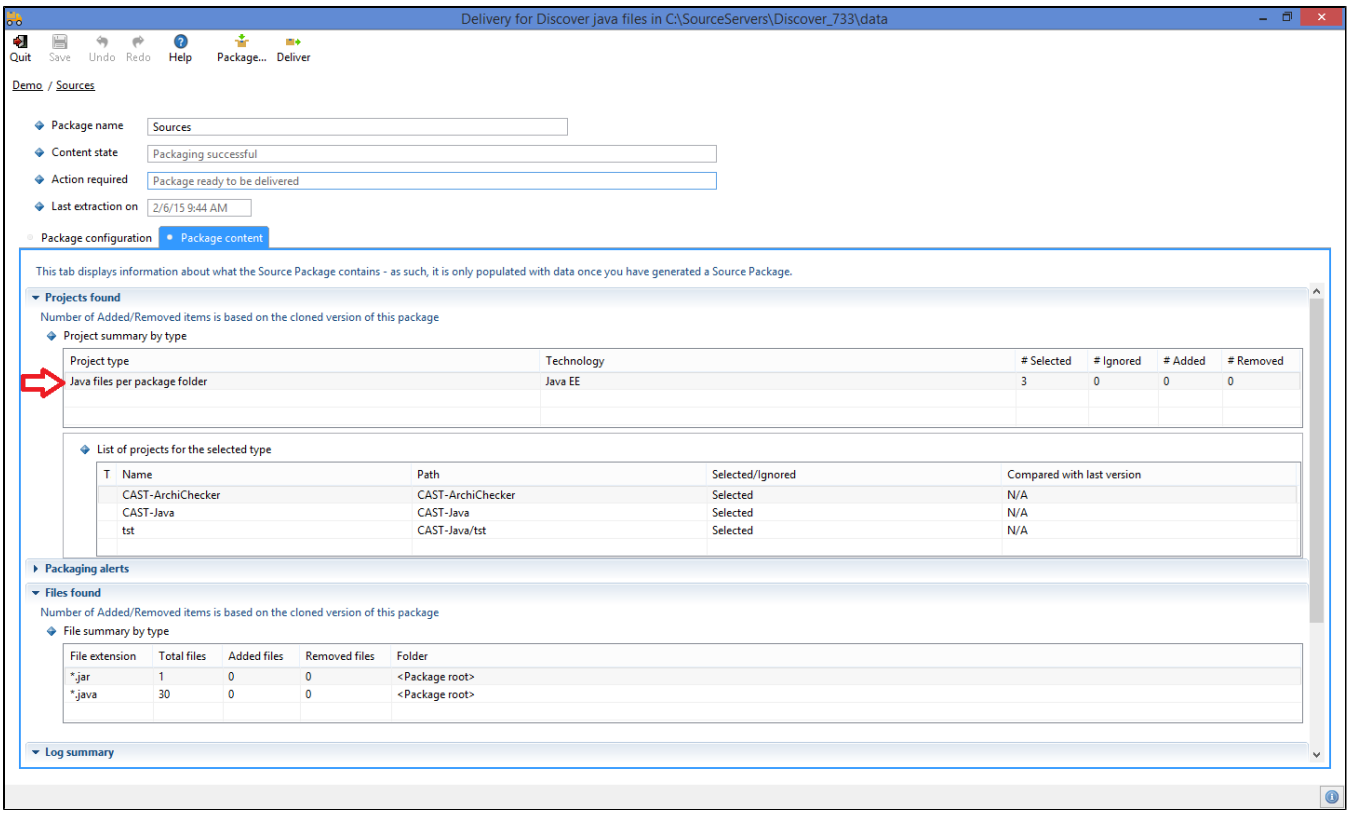
- in the CAST Delivery Manager Tool, in the **Package configuration tab**, the interface contains three **exclusion rules** in the "**Project to exclude**" section - these are all **active by default**:

| | |
|---|---|
| Exclude all 'Java file per package folder' projects when a full JEE project also exists for these files | If the "out of the box" JEE discoverer and the "JEE File Discoverer" identify projects for the same java files, then the "JEE File Discoverer" projects will be ignored in the **Package Content tab** after packaging in favour of the standard "out of the box" JEE discoverer projects. |
| Exclude all 'Java file per package folder' projects with an incomplete package | If the "JEE File Discoverer" identifies projects known as "Incomplete Packages" (as shown in the packaging log), then these will automatically be ignored in the **Package Content tab** after packaging. |
| Exclude all 'Java file per package folder' projects with an unnamed package | If the "JEE File Discoverer" identifies projects known as "Unnamed Packages" (as shown in the packaging log), then these will automatically be ignored in the **Package Content tab** after packaging. |



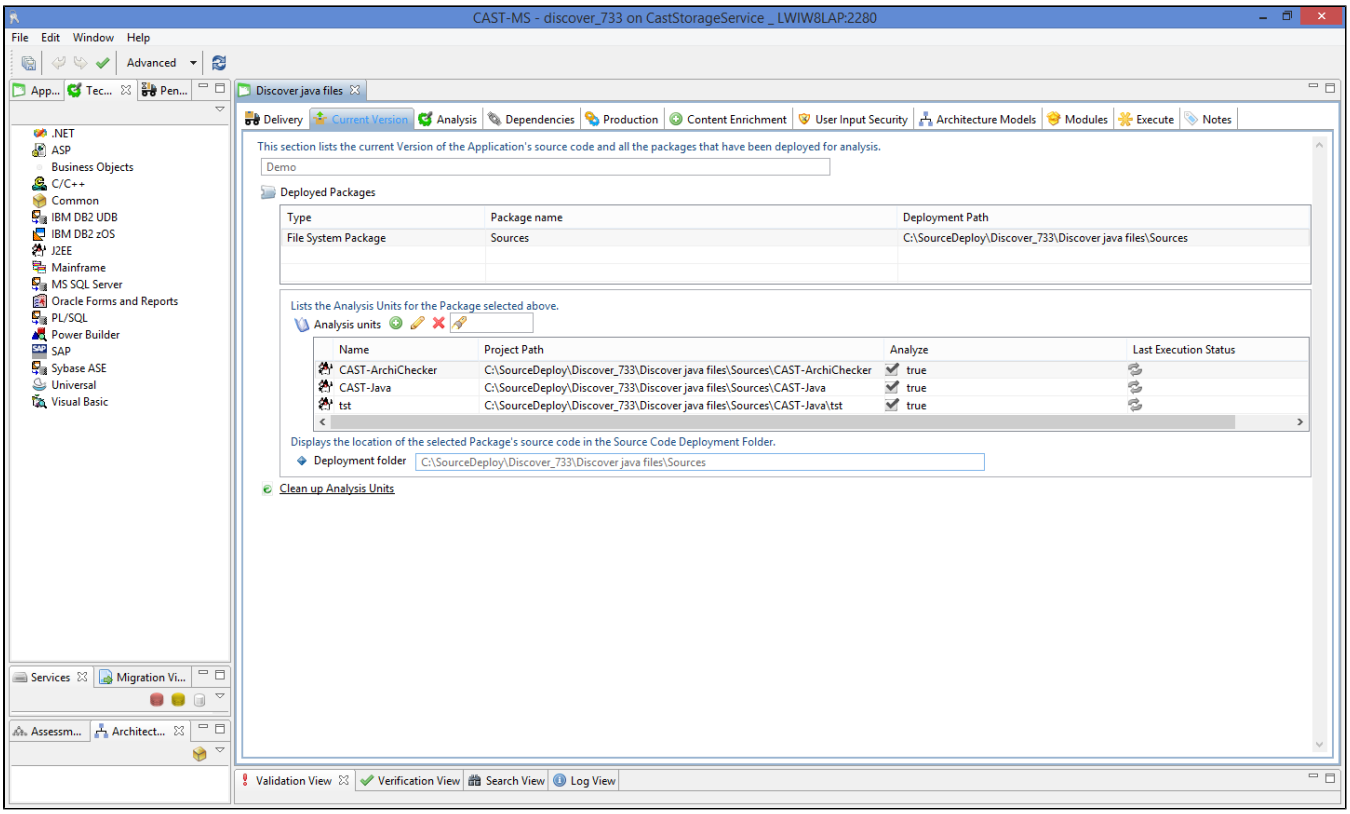

- in the CAST Delivery Manager Tool, in the **Package content tab**, when the package contains some java files, the interface will display a new item in the list of "**Project types discovered**":

*Click to enlarge:*



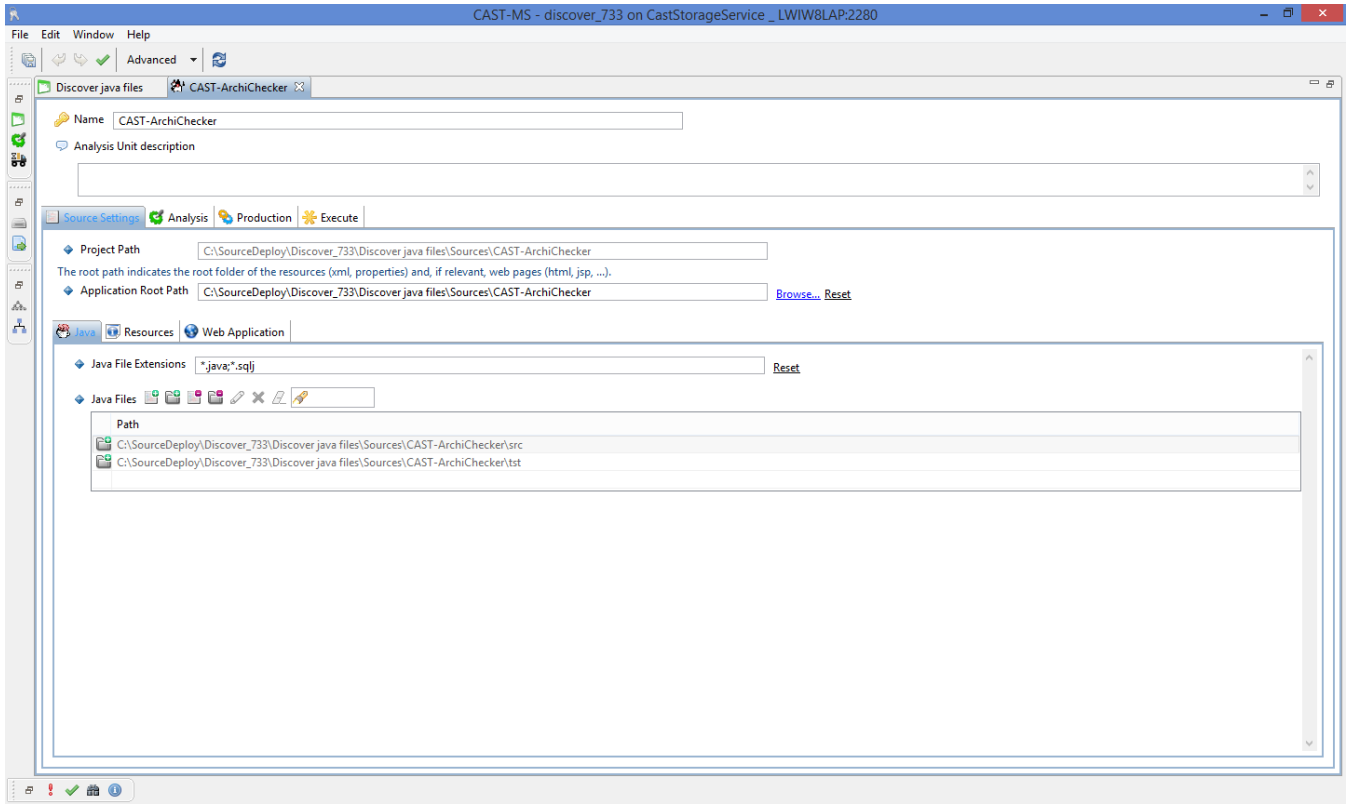

- in the CAST Management Studio, when the delivery is accepted and set as current version, the package will contain Analysis Units corresponding to these projects:

*Click to enlarge:*



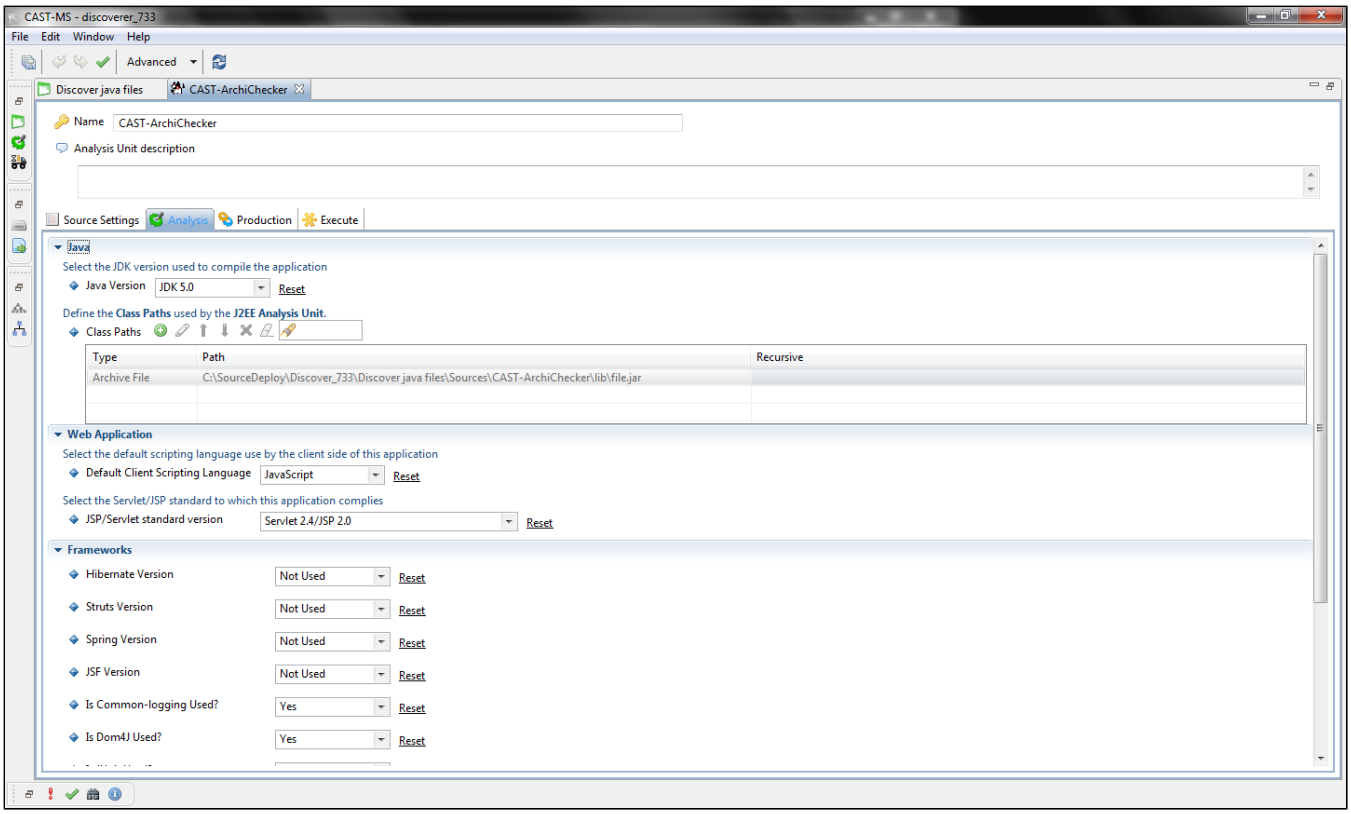

- in the CAST Management Studio, the Analysis Units will contain:
  - the **Source Settings** discovered by the extension:
    - **Application Root Path**
    - **Java Files**: list of parent folders of the main package folder

*Click to enlarge:*



- the **Analysis Settings** discovered by the extension:
  - **Class Paths**: list of jar files inside the Project Path

*Click to enlarge:*

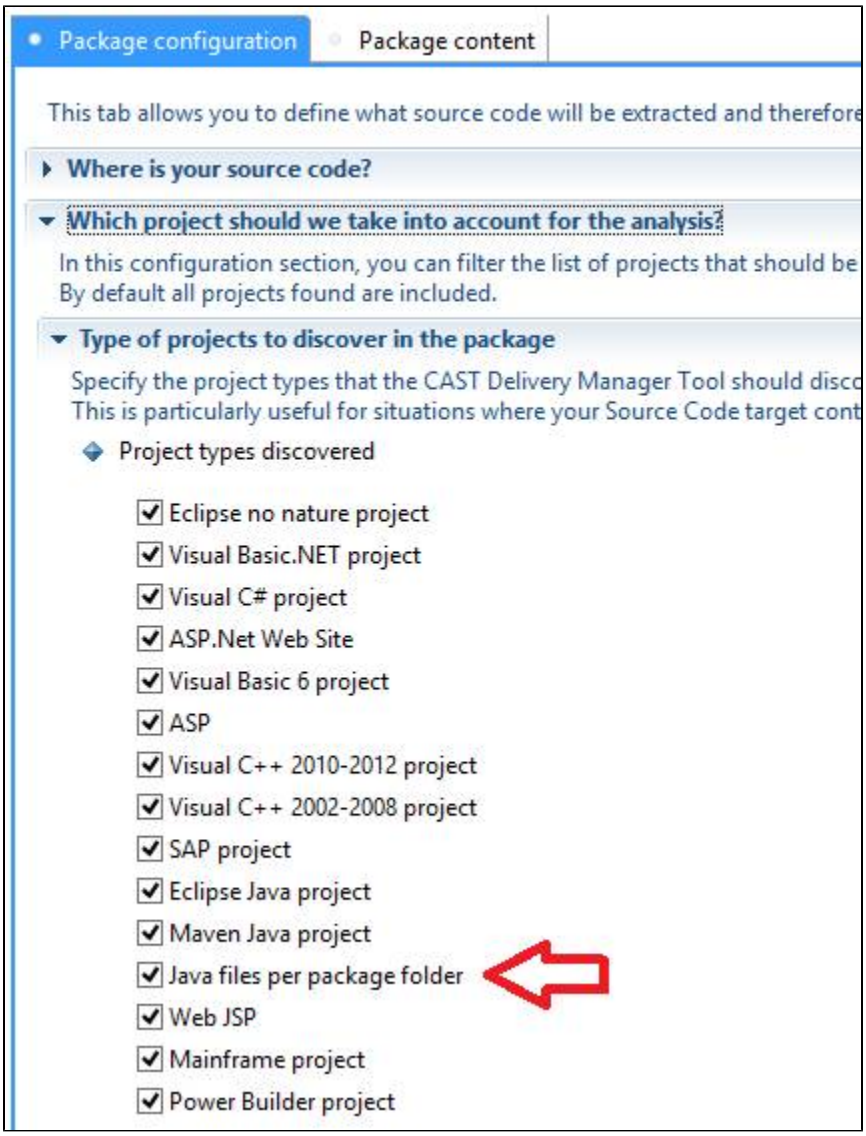



# Packaging your source code with the CAST Delivery Manager Tool

When you use the CAST Delivery Manager Tool to package the source code, please ensure that you tick the following option in the CAST Delivery Manager Tool to **activate the extension**:

Note that:

- this option will be disabled (unticked) in all pre-existing packages - you need to check it before starting the packaging process otherwise it will be ignored.
- for new packages created after the extension's installation, the option will be enabled by default




## Log messages

When packaging your source code and you have activated the JEE File Discoverer, the following messages will be present in the log file to help you understand the nature of the discovery:

> Note that for performance reasons, the extension only scans ONE .java file per folder it encounters, therefore it is highly recommended that if a log message is emitted for a .java file in a specific folder, ALL the .java files in that folder are manually checked consistency.

### Unnamed Package

- **Message type**: INFO
- **Location**: **Scan** and **Discover** phases of the CAST Delivery Manager Tool packaging.
- **Meaning**: The package declaration is missing; therefore, the corresponding Java file should be considered as a "batch" written in Java.

```
Java file %FILE_PATH% has no package, and will lead to a specific JEE file project
```

### Invalid Package

- **Message type**: WARNING
- **Location**: **Scan** phase of the CAST Delivery Manager Tool packaging.
- **Meaning**: The package declaration is incorrect; therefore, the corresponding Java file is skipped from any Java project.

```
Java file %FILE_PATH% has an invalid package, and will be skipped
```

This warning corresponds to an invalid declaration in the java file, for example:

```
package;

package a.b
```

These invalid declarations are typically found in "scrap" .java files, i.e. files which are bundled in the delivery but which are not taken into account during compilation.

## Incomplete Package

- **Message type**: INFO
- **Location**: **Scan** and **Discover** phase of the CAST Delivery Manager Tool packaging.
- **Meaning**: The package declaration does not match the directory structure; therefore, the discovered project might be incomplete.

```
Java file %FILE_PATH% has an incomplete package
```

## Unmatching Package

- **Message type**: INFO
- **Location**: **Scan** phase of the CAST Delivery Manager Tool packaging.
- **Meaning**: The package declaration does not **at all** match the directory structure; therefore, the discovered project might be incomplete.

```
Java file %FILE_PATH% has an unmatching package
```

## Parent Project

- **Message type**: INFO
- **Location**: **Discover** phase of the CAST Delivery Manager Tool packaging.
- **Meaning**: The folder associated to the project contains sub folders associated with valid (i.e., they are not Invalid Packages, Incomplete Packages or Unmatching Packages) projects; in this situation, some XML and properties resource references are added to the configuration, so that the application root path, which is too large, will not be used as a default by the analyzer to scan the XML and properties files.

```
Java project %PROJECT_NAME% under %PROJECT_PATH% contains additional Java projects
```