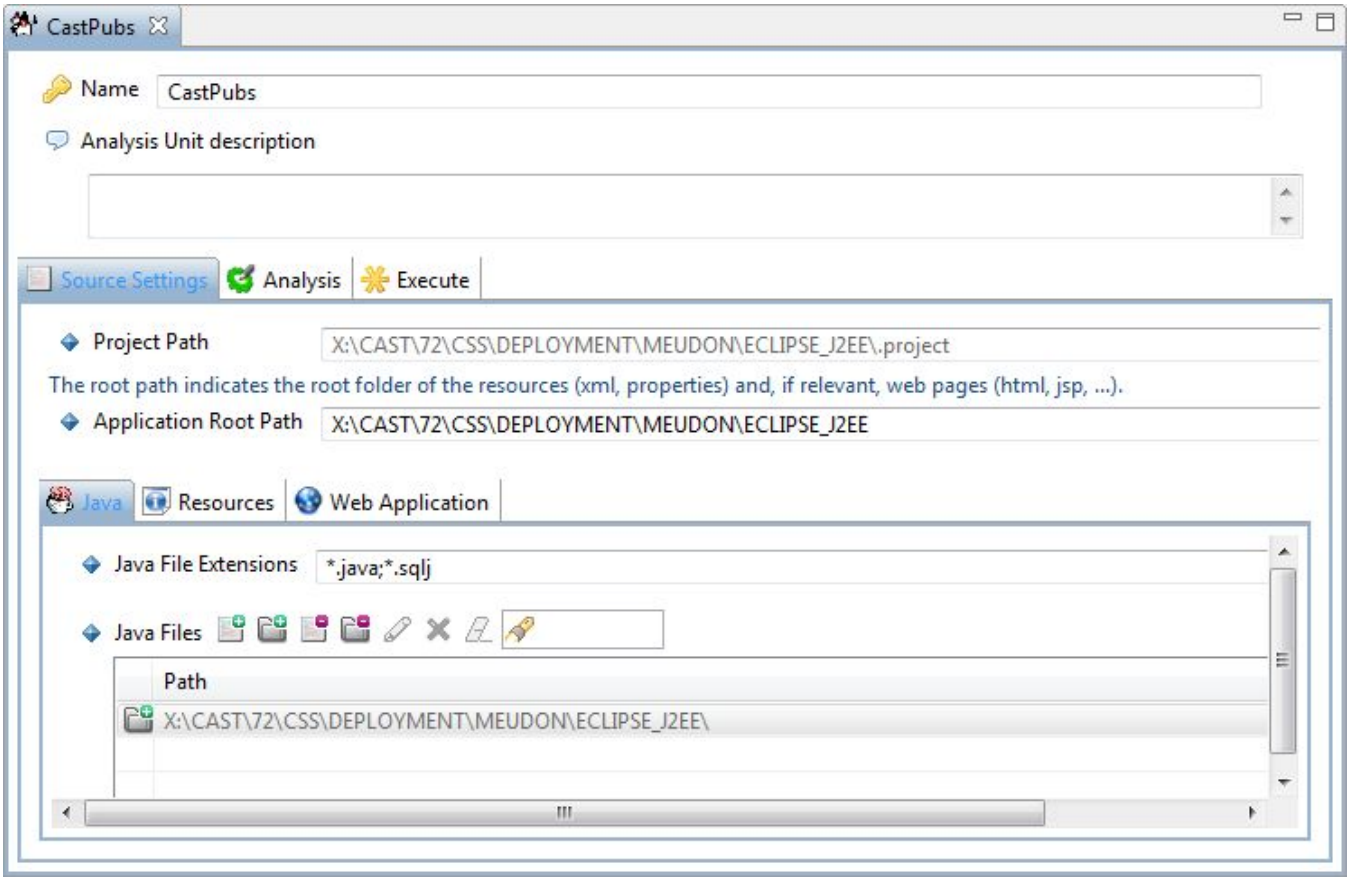
Review Technology and Dependency settings

On this page:

Summary: Confirming the analysis settings requires a review of technology and dependency settings.

Technology settings

The packages delivered through the DMT are automatically transformed into Analysis Units where project files exist. CAST AIP will automatically detect the type of source code and recommend the most appropriate analysis settings for each Analysis Unit, for example:



Analysis settings configured at the Analysis Unit level override any configuration at the Application level which in turn override any configuration at the Technology level (in the CAST Management Studio).

If any change in the analysis configuration is required, for example to address a warning discovered in the analysis log, that change must be made in the analysis configuration that pertains to the specific Analysis Unit. The same change should be made to each Analysis Unit if it pertains the entire applications because of the configuration override precedence mechanism explained here.

The Analysis tab in the Analysis Unit editor should be used very carefully when it is impractical to set the configuration across multiple Analysis Units and you need to set a configuration for many Analysis Units at the same time. In this case you must ensure that no unexpected override configuration take place at the Analysis Unit level.

CAST's recommended best practice is to accept the suggested configuration and make any changes only retroactively to address analysis log warnings and/or errors. If the application is very big, complex or you have to make some manual changes to address unsupported conditions, you could run the Analysis in Test Mode (analysis results are not saved) and fix any issues iteratively.

For more details on auto configuration, current limitations and additional validations that should be performed by the CAST AI Admin see the following technology specific pages:

- ABAP - confirm analysis settings
- C and Cpp - Confirm analysis configuration
- Mainframe - Confirm analysis configuration

If you are using CAST AIP extensions, you may also need to consult their documentation.

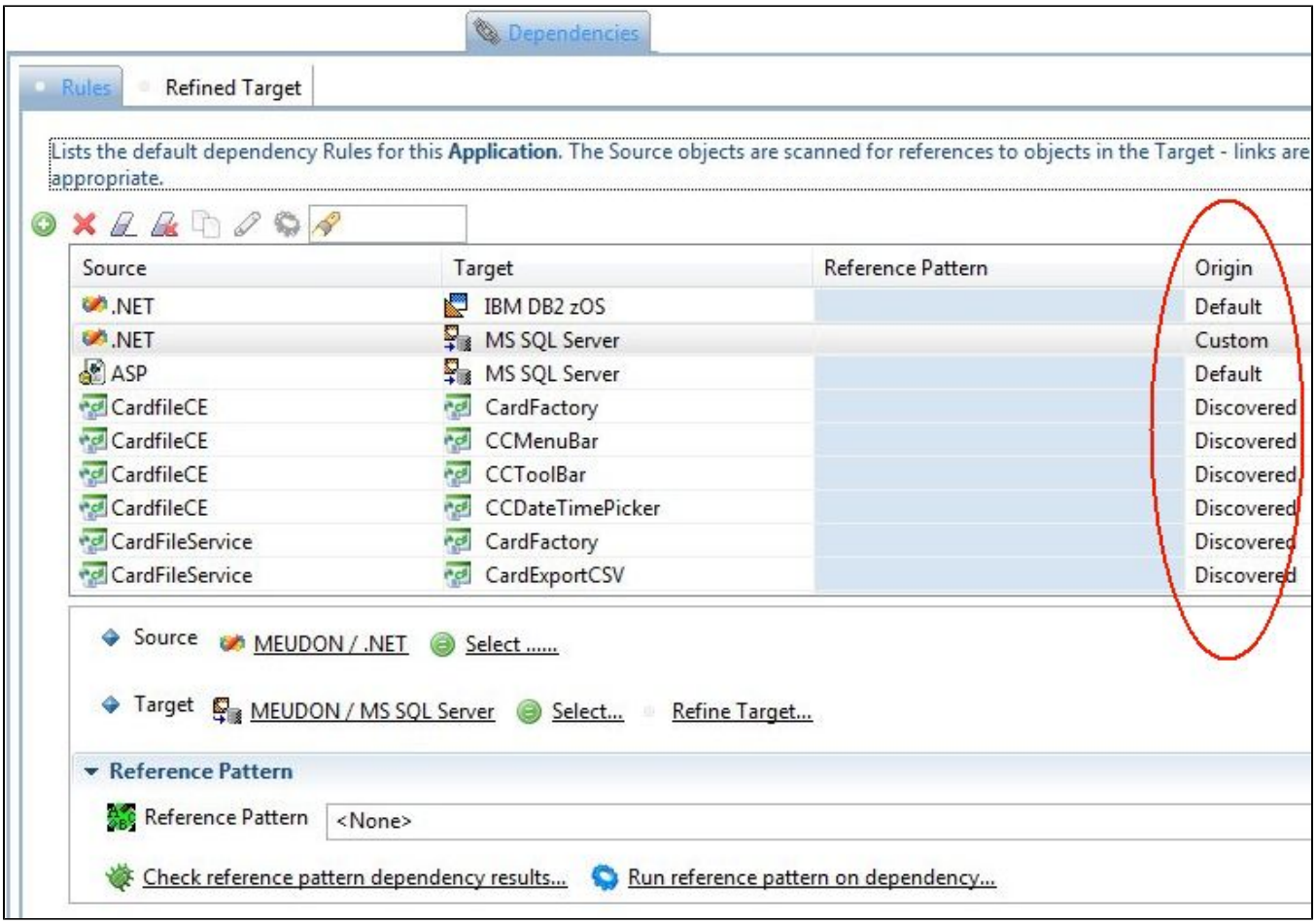
Dependency settings

This step involves checking that the dependencies settings are correct for your technologies. Dependencies are configured at Application level in the **Dependencies tab**.

When a dependency exists, during the analysis process source code corresponding to the Technologies/Analysis Units in the **Source** column is scanned for references to objects in source code corresponding to the Technologies/Analysis Units in the **Target** column. If any matches are found, then a link between the two objects will be created (this will be visible in CAST Enlighten for example).

References are traced using **search strings** which is less selective than parser based technology used for other links traced by the analyzer. This technology detects a reference to an object wherever its name is mentioned, regardless of the context in which this reference occurs. As a result, **incorrect links** may be traced if a string happens to match a given name even if it is not logically related to the corresponding object. As a result you may have to **intervene** to filter incorrect references - see Validate Dynamic Links.

There are **three types of dependencies**, explained below:



Incorrect dependency settings may result in missing links - missing inter-technology dependencies - or in a large number of Dynamic Links - incorrect dependencies. This can have a knock on effect on Quality Rule results and on Transaction flow.

Default rules

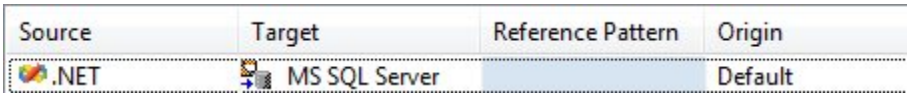
Default dependency Rules are created **automatically** by the CAST Management Studio between Technologies, as follows:

If no dependency Rules are "discovered" (during the package configuration in the CAST Discovery Manager Tool) between Analysis Units in the Application, then CAST will use a "default" rule defined at Technology level. Note that generally speaking **database technologies** do not have any technologies set as **Target**. The exception to this is MS SQL Server which has a default Target .NET technology.

The **default rules** are as follows:

- any "client" results depend on all "server" analysis results
- JEE analysis results depend on Mainframe results

In the example screen shot below, no dependencies were "discovered" between any .NET Analysis Units and MS SQL Server Analysis Units - as such a default Technology level rule was used to ensure links were created where references exist:



A more complex example of this feature is as follows. Take a situation where we have the following rules defined at Technology level:

- JEE > JEE
- JEE > SQL
- JEE > Mainframe

If the following Analysis Units exist in the Application:

- AU1 (JEE)
- AU2 (JEE)

- AU3 (JEE)
- AU4 (SQL)

And the CAST Delivery Manager Tool has discovered the following dependencies automatically:

- AU1 > AU2
- AU3 > AU2

Then the following dependencies will exist in this tab:

- AU1 > AU2
- AU3 > AU2
- JEE > SQL

In other words, the JEE > JEE rule has been removed because CAST has discovered that certain Analysis Units depend on others and has created specific Discovered rules for them. No dependencies were discovered between any JEE Analysis Units and SQL Analysis Units, therefore the default rule JEE > SQL will be applied. The JEE > Mainframe rule will not appear because there are no Analysis Units of this Technology type in the Application.

Discovered rules

Discovered dependency Rules are also created **automatically** by the CAST Management Studio - these Rules are directly "discovered" from the source code by the **CAST Delivery Manager Tool**. Dependency Rules that have been "discovered" are always **between Analysis Units** since they are based on dependencies between projects (i.e. a classpath, for example for JEE, .NET, C/C++ etc.):

Source	Target	Reference Pattern	Origin
CardfileCE	CardFactory		Discovered
CardfileCE	CCMenuBar		Discovered
CardfileCE	CCToolBar		Discovered
CardfileCE	CCDateTimePicker		Discovered
CardFileService	CardFactory		Discovered

If there were no "missing project" alerts in the CAST Delivery Manager Tool, then you can be confident that all "discovered" dependencies are correct. If you do have "missing project" alerts in the CAST Delivery Manager Tool, you should always try to fix these first and if this is not possible you can then create a custom dependency where the projects are available.

Custom rules

Custom dependency Rules are those that have been created manually. Typically these are created to cover a situation which the automatic Default and Discovered dependency Rules do not take into account - you will need to examine the source code manually to determine what dependencies you should create.

One example where a custom rule might be necessary is in **cross technology situations** where code in Analysis Unit Technology A relies on code in Analysis Unit Technology B:

Source	Target	Reference Pattern	Origin
CardFileService	APT320		Custom

Another example would be where a "discoverer" **does not exist for a technology** and you are forced to create User Defined Analysis Units - in this case dependency rules must be created manually.

Note that:

- although it is possible to add a custom dependency with a technology (meaning "All Analysis Units of the given technology"), you should be aware that this will create dynamic links (see Validate Dynamic Links) and may also impact analysis performance.
- if you define custom dependencies with technologies, the CAST Management Studio will remove any default rules covering the same technologies: you are now responsible for managing the dependencies.

Where several client technologies (for instance Java and C/C++) or several database schemas exist, the CAST AI Admin must set the correct dependencies, overriding the default settings if necessary. The fine tuning of dependencies settings can be configured by rules in the CAST_INSTALL_FOLDER\configuration\parametrization**CustomExternalLinksRules.xml** file - please see the CAST Management Studio help for more information about this.