PL1 1.0

On this page:

Target audience:

Users of the extension providing **Enterprise PL/I for z/OS** support.

Summary: This document provides information about the extension providing **Enterprise PL/I for z/OS** support.

What's new?

Please see PL1 1.0 - Release Notes for more information.

Description

This extension provides support for applications written using **Enterprise PL/I for z/OS** languages.

In what situation should you install this extension?

If your application contains source code written using **PL/I** and you want to view these object types and their links with other objects, then you should install this extension.

Supported Versions of PL/I

Although this extension is officially supported by CAST, please note that it has been developed within the technical constraints of the CAST Universal Analyzer technology and to some extent adapted to meet specific customer needs. Therefore the extension may not address all of the coding techniques and patterns that exist for the target technology and may not produce the same level of analysis and precision regarding e.g. quality measurement and/or function point counts that are typically produced by other CAST AIP analyzers.

This version of the extension provides support for:

Enterprise PL/I for z/OS	Supported
5.x	✓
4.x	✓

3.x	✓

Function Point, Quality and Sizing support

This extension provides the following support:

- **Function Points (transactions)**: a green tick indicates that OMG Function Point counting and Transaction Risk Index are supported
- **Quality and Sizing**: a green tick indicates that CAST can measure size and that a minimum set of Quality Rules exist

Function Points (transactions)	✓
Quality and Sizing	✓

CAST AIP compatibility

This extension is compatible with:

CAST AIP release	Extension release	Supported
8.3.x	1.0.1	✓
8.2.x	1.0.1	✓
8.1.x	1.0.1	✓
8.0.x	1.0.1	✓
7.3.4	1.0.1	✓

Supported DBMS servers

DBMS	Supported?
CSS	✓
Oracle	✓
Microsoft SQL Server	✗

Prerequisites

✓	An installation of any compatible release of CAST AIP (see table above)

Download and installation instructions

Please see:

- Download an extension
- Install an extension

> The latest release status of this extension can be seen when downloading it from the CAST Extend server.

Prepare and deliver the source code

Once the extension is downloaded and installed, you can now package your source code and run an analysis. The process of preparing and delivering your source code is described below:

Source code preparation

Only the following **file extensions** are recognised by the CAST Delivery Manager Tool:

- .PLC
- .PLI

However, PL/I source code files can sometimes be created with any extension (for example .TXT) or no extension at all. If these type of files are encountered by the CAST Delivery Manager Tool during the packaging action, it will try to determine whether the file is a valid PL/I source file and if so, it will change the extension to .PLI or .PLC (for files with extensions other than .PLI or .PLC) and will add a .PLI or .PLC extension for files with no extension.

Currently, the PL/I extension ignores **empty files** and they will not be considered as valid source code files.

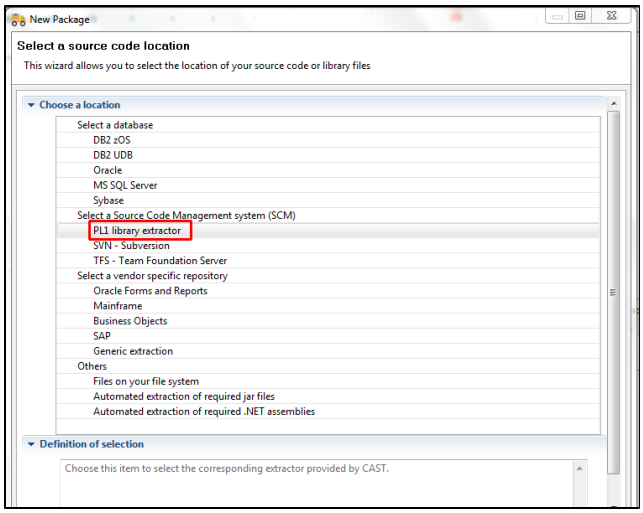
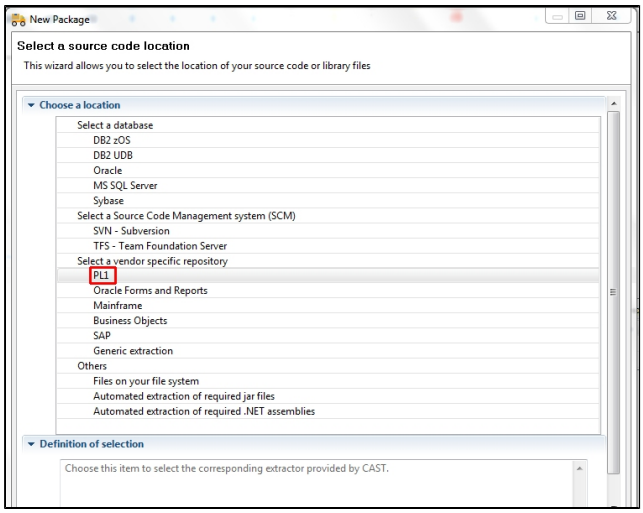
Binary files should also be avoided since it is not always possible to detect whether a given file is binary or not. Presenting binary files to the CAST Delivery Manager Tool will **not** result in a packaging failure, but an error message can be expected in the packaging/extraction log.

Deliver the source code

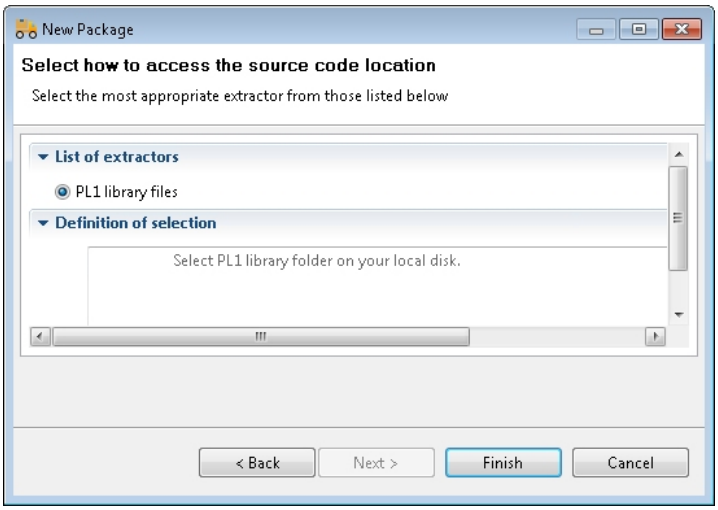
Using the CAST Delivery Manager Tool:

- create a new **Version**
- create a new **Package** for your PL/I source code as follows:

PL1 1.0.3

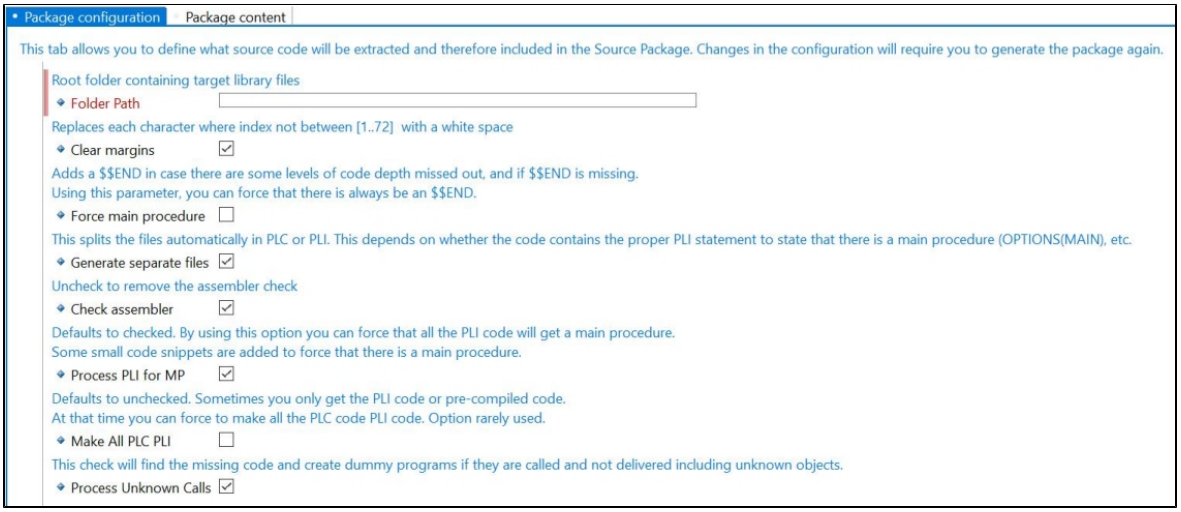
PL1 1.0.3	PL1 1.0.4
Choose the option **PL1 Library Extractor** in the **SCM** section as the source location: *Click to enlarge:* 	Choose the option **PL1** in the **vendor specific repository** section as the source location: *Click to enlarge:* 

- Now choose the **PL1 Library Files** option and click **Finish**:



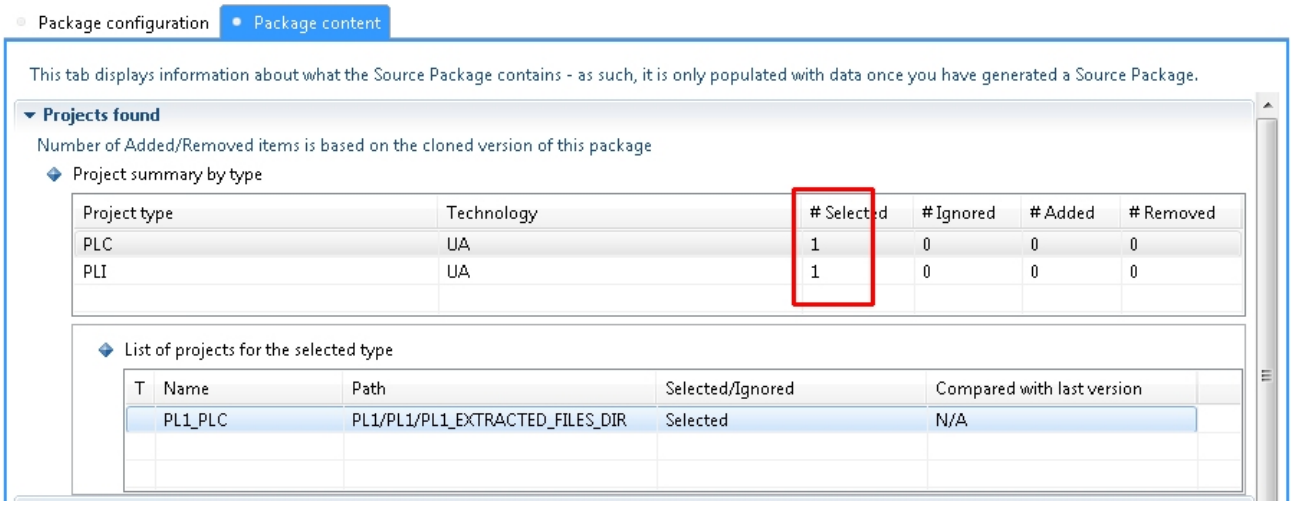
- Now **configure** the Package in the **Package configuration tab** - at a minimum you need to specify the **location** of your target PL1 library files in the **Folder path** field (for example a location on disk). All other options can be left at their default unless you specifically need to change them:

Click to enlarge:



- Run the **Package action**: the CAST Delivery Manager Tool will **discover** any "projects" related to the PL1 application source code:

Click to enlarge:

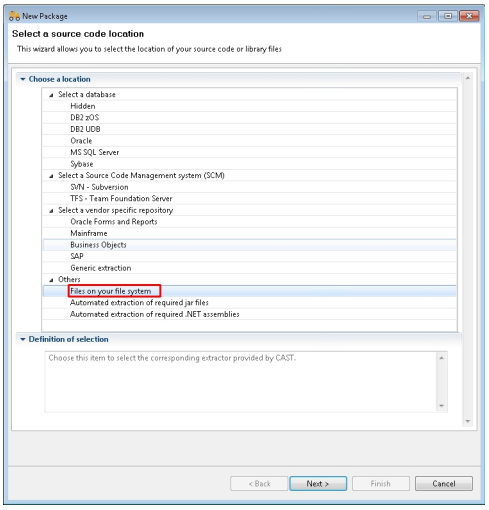


- Finally, deliver the **Version**

PL1 1.0.2

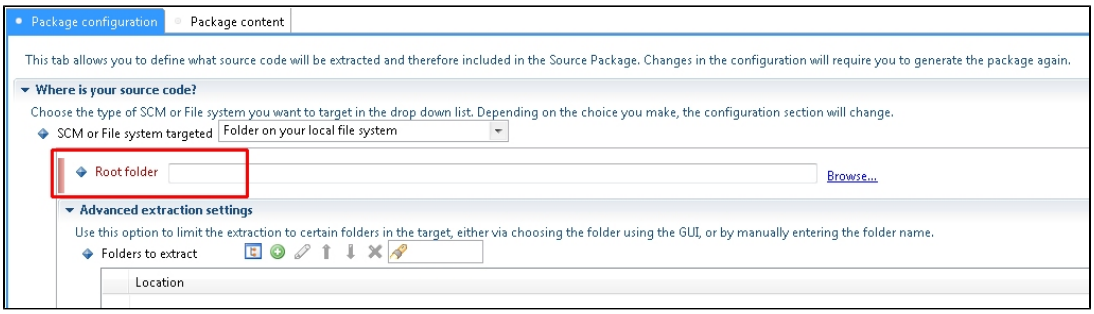
- Choose the **Files on your file system** option:

Click to enlarge:



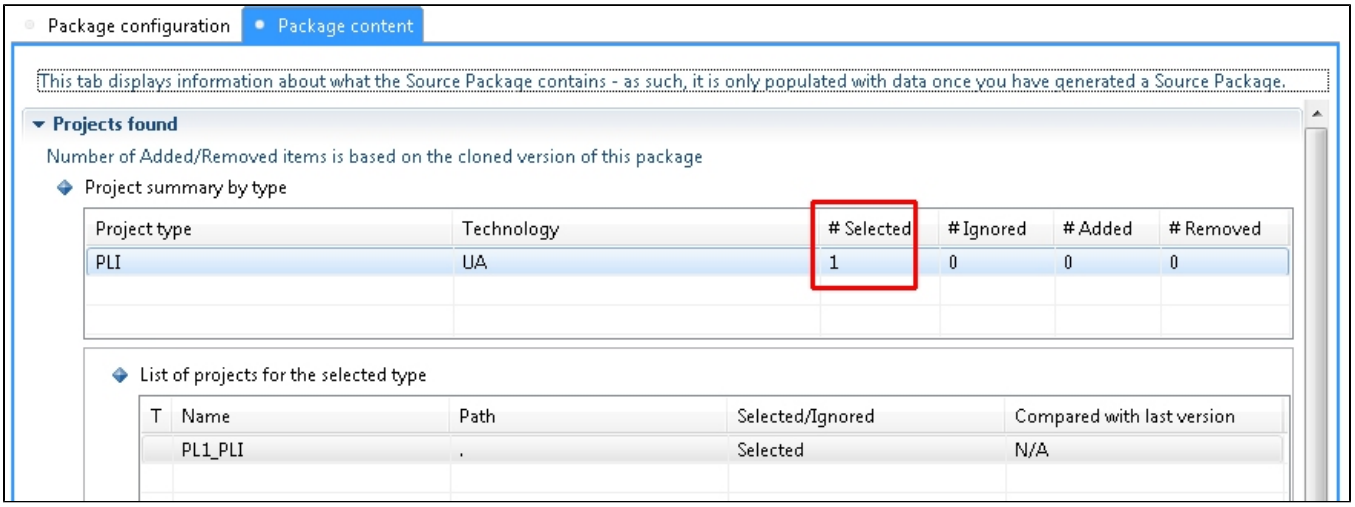
- Now **configure** the Package in the **Package configuration tab** - at a minimum you need to specify the **location** of your target PL1 library files in the **Root folder** field (for example a location on disk):

Click to enlarge:



- Run the **Package action**: the CAST Delivery Manager Tool will **discover** any "projects" related to the PL1 application source code:

Click to enlarge:



- Finally, deliver the **Version**

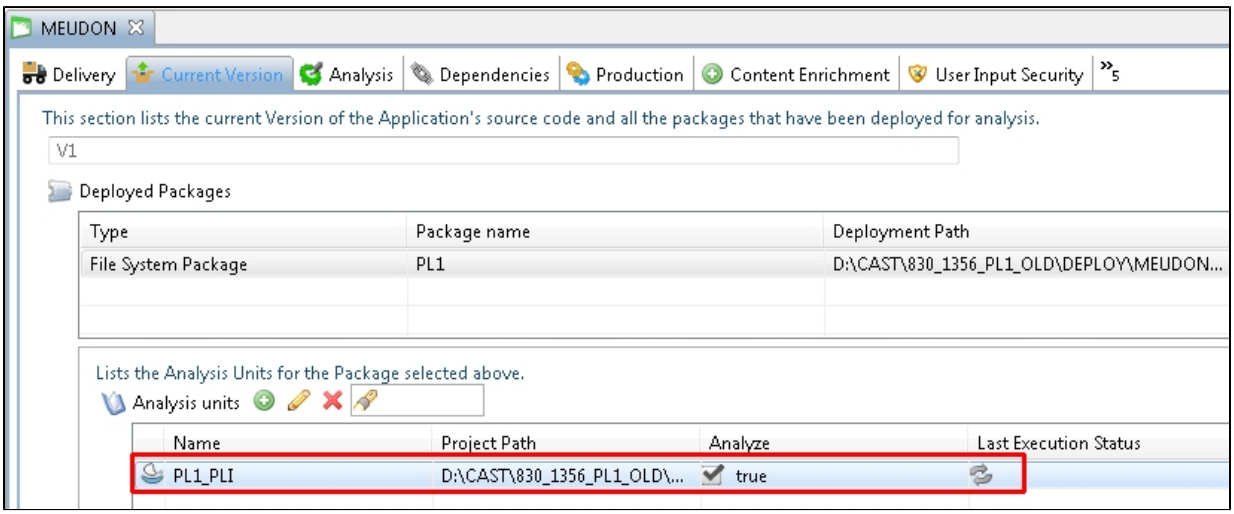
Analysis configuration and execution

Refer to Analysis Configuration and Execution for more information.

Using the CAST Management Studio:

- Accept and deploy the **Version** in the CAST Management Studio. **Analysis Units** will be created automatically relating to the PL/I source code:

Click to enlarge:



- Run a **test analysis** on the Analysis Unit before you generate a **new snapshot**.

Logging mechanism

Analysis log files

Analysis logs are stored in the default locations used by the CAST Management Studio.

Error/Warning Message	Action
Unable to find the object end for type 'XXXXXXX'	This is caused by complex source code that makes the engine miss the end of the object, whereas the begin of object type was captured. Object won't be created in Analysis Service, will lead to incomplete analysis. Issue must be reported to CAST Support, along with source code (original source file + preprocessed source code) to allow reproduction and investigation. A few such warnings for object types like PLIProcedure, PLCProcedure, PLIFunction could be acceptable. Same warning for PLIMainProcedure is problematic : the whole program is not saved in Analysis Service.
An unnamed object of type 'PLIWhenCall' has been detected	This is because PLIWhenCall has no specific name, as such, the CAST framework will give them the name "unnamed". You can safely ignore this warning.
• Duplicate object of type 'PLIWhenCall' has been detected: '<unnamed>' • Duplicate object of type 'PLIErrorBlock' has been detected	As mentioned above, PLIWhenCall and PLIErrorBlock have no specific name and are automatically given the name "unnamed" by CAST - as such multiple objects with the name "unnamed" will exist and will cause this error. You can safely ignore this warning. Same for PLIProcSubscriptedVar object type.
• Duplicate object of type 'PLIProcedure' has been detected • Duplicate object of type 'PLCProcedure' has been detected	PLIProcedure and PLCProcedure are supposed to be unique within the same source file. So such warning is the sign of an incorrect object type identification. Issue must be reported to CAST Support.

• end of string '"' not found • end of string '[']' not found • File skipped	UA engine did not find the end of a string, whereas the begin of the string was captured. This generally leads to a second warning : File skipped, meaning the source file and contained objects have not been saved in Analysis Service. On large applications, a few such warnings could be acceptable. Otherwise, issue must be reported to CAST Support, along with source code (original source file + preprocessed source code) to allow reproduction and investigation.
end of comment '[*][/]' not found	Alone, this warning can be safely ignored. Check if other warnings are present for the same source file.
• Could not calculate code depth correctly for file File: <file path> • Could not calculate code depth correctly for file, added <TAG> File: <file path> where TAG can be $END or $$END.	The analyzer is not capable of detecting the source code depth correctly.
PL1 Extractor Version: <version>	A simple information message to help identify the version of the extractor that is being used.

PL/I Preprocessor

PL/I source code needs to be preprocessed so that CAST can understand it and analyze it correctly. This source code preprocessing is **actioned automatically.** In other words you only need to package, deliver and launch an analysis/generate a snapshot for the preprocessing to be completed.

Note that as part of the source code pre-processing phase, from PL1 1.0.5 onwards, CAST will now add "**$$**" to all PL/I source code just prior to the "**;**". For example:

Original source code	`someproc:proc;`
Source code after pre-processing in 1.0.5	`someproc:proc$$;`

The impact of this change is as follows:

- When upgrading to PL1 1.0.5, existing objects will be shown as **updated** when a post upgrade snapshot is run.
- There will be a **reduction** in the number of messages of the type "end of object of type <PLC\PLIProcedure|Function> not found" that were previously displayed in the log.

This change has been implemented because of a limitation in the analyzer with regard to the way object **start and end patterns** are handled. In PL/I, object start and end patterns do not match (contrary to most other languages, such as PHP where { and } are used) and therefore the analyzer is not able to correctly identify when an object ends.

What results can you expect?

Objects

PL/I

Icon	Metamodel description
	PLI DB2 Table
	PLI DO Variable
	PLI Error Block
	PLI FileStructure

	PLI Function
	PLI MainProcedure
	PLI Procedure
	PLI Project
	PLI Subscripted Variable
	PLI When CALL constructs

PLC

Icon	Metamodel description
	PLC DB2 Table
	PLC FileStructure
	PLC Function
	PLC Include
	PLC MainProcedure
	PLC Subscripted Variable
	PLC Procedure
	PLC Project
	PLC TDLI Call

Structural Rules

The following structural rules are provided:

1.0.6	https://technologies.castsoftware.com/rules?sec=srs_pl1&ref=||1.0.6
1.0.5	https://technologies.castsoftware.com/rules?sec=srs_pl1&ref=||1.0.5
1.0.4	https://technologies.castsoftware.com/rules?sec=srs_pl1&ref=||1.0.4
1.0.3	https://technologies.castsoftware.com/rules?sec=srs_pl1&ref=||1.0.3

1.0.2	https://technologies.castsoftware.com/rules?sec=srs_pl1&ref=||1.0.2

You can also find a global list here:

https://technologies.castsoftware.com/rules?sec=t_1004000&ref=||