Review Dynamic Links

On this page:

- Overview
- Which technologies are affected?
- How to manually inspect Dynamic Links
- Automated Dynamic Link review
- Technical notes
    - Limitation of the DLM source code viewer for C/C++
        - Viewing Source Code
        - Macros

Target audience:

CAST Administrators

Overview

Ensuring the correct discovery of transactions is a multi-step process that may be complex and time consuming for applications that are not supported out-of-the-box. At the core of CAST transaction discovery algorithm is the understanding of the links between objects discovered during the source code analysis of the target application.

CAST AIP automatically sets up default Dependency Rules when the source code is delivered and set as the current version: the source code objects are scanned for references to other objects and links are created where appropriate. Not all links generated in this fashion are valid and their validation by the CAST AI Admin is therefore required.

Missing links result from the inability to find references between objects and can also occur either because of missing dependencies or because of the presence of frameworks that are not supported out-of the box via an environment profile. In this case, new links may be added by defining new dependencies, custom association rules (using the CAST Reference Pattern tool) and/or new, custom environment profiles developed ad hoc to extend the out-of the box support.

For cross-technology links, External Links will identify and record a link between two objects whose validity cannot be precisely determined. These links are tagged as "**dynamic**". The CAST AI Admin's inspection of these dynamic links is necessary to determine whether the link in question is legitimate (i.e. valid) or if instead it should be ignored and removed from the Analysis Service.

Why does it matter?

Inspection of dynamic links is a ***mandatory step***.

It is a very important step because it impacts the following diagnostic results in the CAST Engineering Dashboard :

- Architecture Checker rules (Architecture Model)
- High Fan-in artifacts (Dependencies)
- High Fan-out artifacts (Dependencies)
- Unreferenced components (dead code)
- Other architecture rules

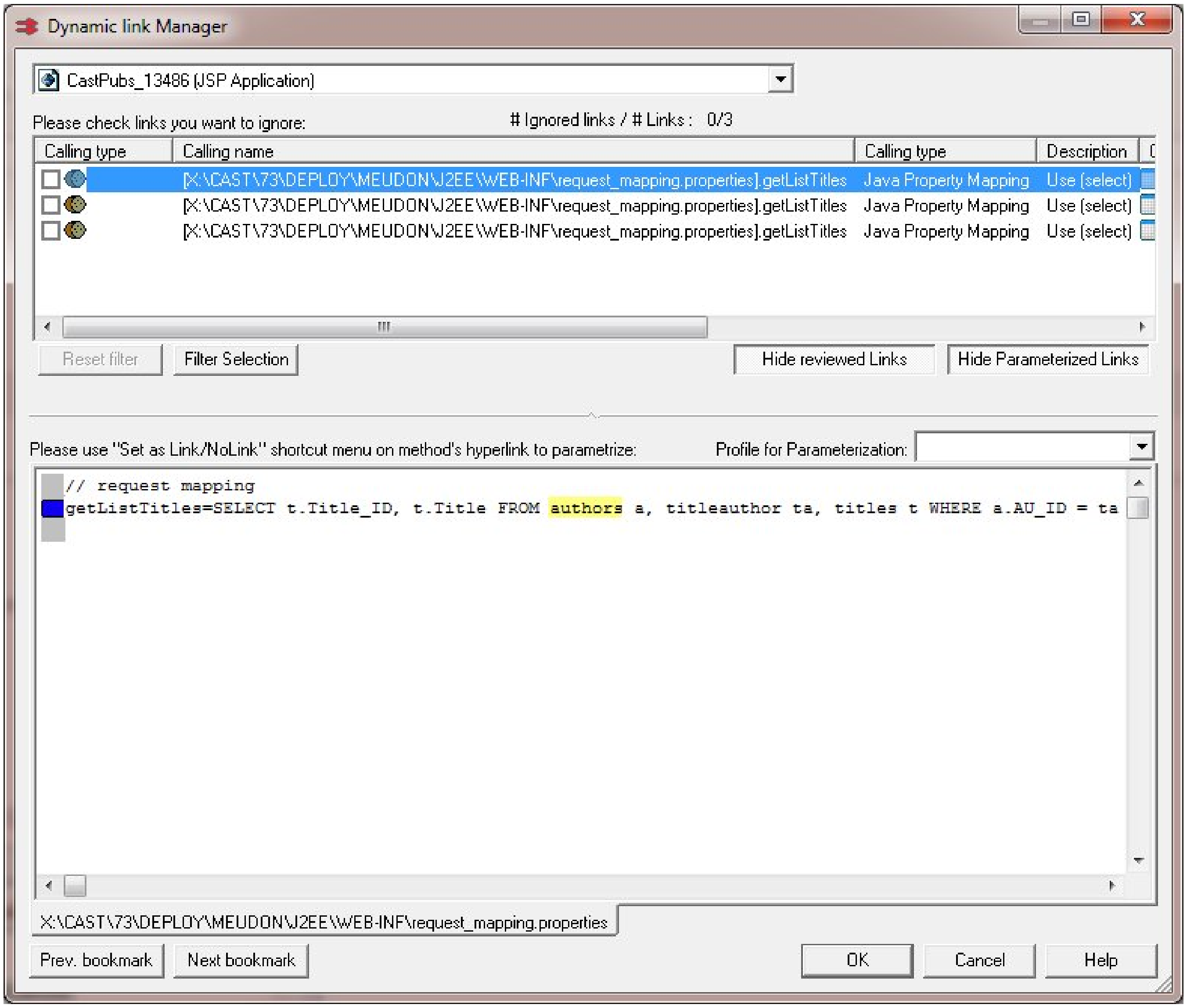
Which technologies are affected?

- J2EE
- New .NET
- C, C++.

COBOL and Pro*C are less affected, since embedded SQL will produce plain links.

How to manually inspect Dynamic Links

The Dynamic Link Manager (DLM) tool is accessible from CAST Management Studio and must be used to review the Dynamic Links in the first instance:


Dynamic link Manager
CastPubs_13486 (JSP Application)
Please check links you want to ignore:
# Ignored links / # Links : 0/3

Calling type	Calling name	Calling type	Description
☐	[X:\CAST\73\DEPLOY\MEUDON\J2EE\WEB-INF\request_mapping.properties].getListTitles	Java Property Mapping	Use (select)
☐	[X:\CAST\73\DEPLOY\MEUDON\J2EE\WEB-INF\request_mapping.properties].getListTitles	Java Property Mapping	Use (select)
☐	[X:\CAST\73\DEPLOY\MEUDON\J2EE\WEB-INF\request_mapping.properties].getListTitles	Java Property Mapping	Use (select)

Reset filter
Filter Selection
Hide reviewed Links
Hide Parameterized Links
Please use "Set as Link/NoLink" shortcut menu on method's hyperlink to parametrize:
Profile for Parameterization:

```
// request mapping
getListTitles=SELECT t.Title_ID, t.Title FROM authors a, titleauthor ta, titles t WHERE a.AU_ID = ta
```

X:\CAST\73\DEPLOY\MEUDON\J2EE\WEB-INF\request_mapping.properties
Prev. bookmark
Next bookmark
OK
Cancel
Help

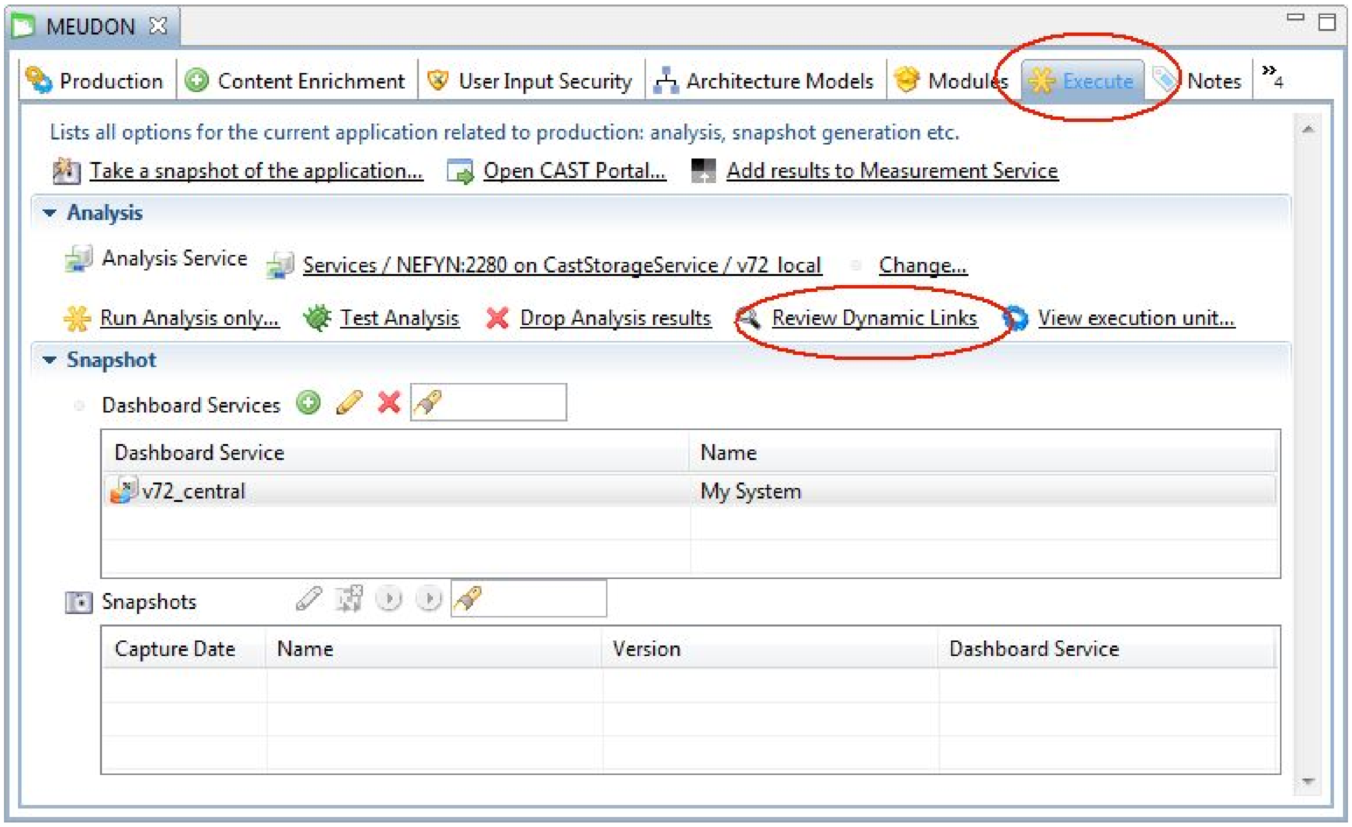
- **Invalid** links must be "ignored" after reviewing the code that creates the link (by ticking the box next to link)
    - **Ignore** Dynamic Links when:
        - Source code uses a logger information message
        - Basic text displayed in a message box or a frame
- **Valid** links must be "validated" after reviewing the code that creates the link (by right clicking the link and selecting Validate)
    - **Validate** Dynamic Links when:
        - code uses a manipulation of SQL queries
        - code uses a direct call from a business layer to a database table or function
        - code links a client part and a server part of an application

When in doubt, ignore Dynamic Links not resolved automatically (see below) through parameterization / filter rules or via manual selection.

Note that:

- all Dynamic Links are by default assumed as if they were confirmed and thus included in the Analysis Service.
- you can close the Dynamic Links Manager in the middle of a review session and changes will persist.

These actions will remain valid for the next re-analysis of same artifact. You can access the DLM from the CAST Management Studio **Application editor** (Execute tab):



MEUDON

Production | Content Enrichment | User Input Security | Architecture Models | Modules | Execute | Notes

Lists all options for the current application related to production: analysis, snapshot generation etc.

Take a snapshot of the application... | Open CAST Portal... | Add results to Measurement Service

Analysis

Analysis Service | Services / NEFYN:2280 on CastStorageService / v72_local | Change...

Run Analysis only... | Test Analysis | Drop Analysis results | Review Dynamic Links | View execution unit...

Snapshot

Dashboard Services

Dashboard Service	Name
v72_central	My System

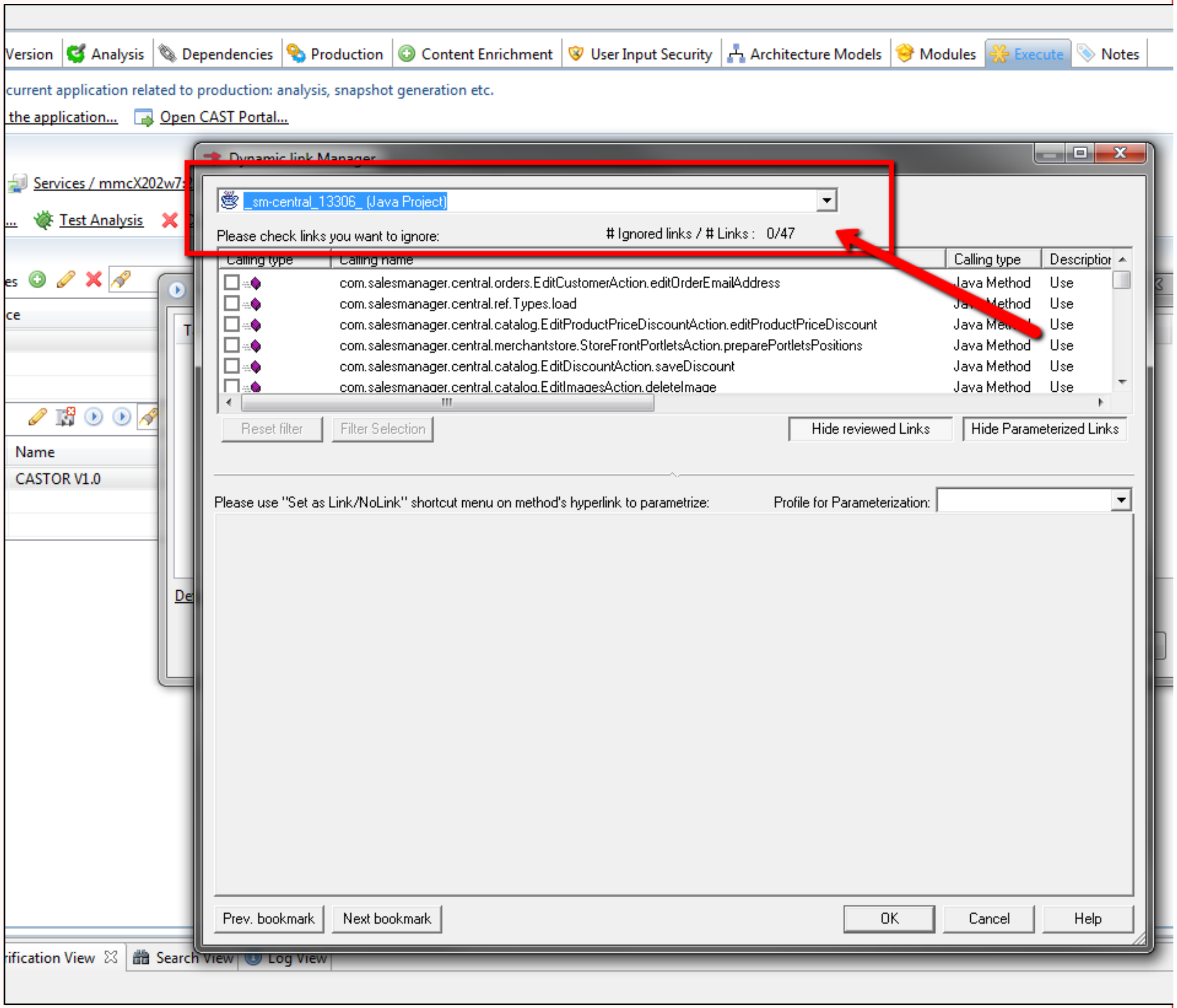
Snapshots

Capture Date	Name	Version	Dashboard Service

Warning

The Dynamic Link Manager shows dynamic links grouped by Analysis project (one or more for each Analysis Unit in a standard configuration) **fo r all** Applications analyzed with the CAST Management Studio. As such, the list displayed may not refer to the Application you are working on. Additionally, there is no ability to filter the specific application you are working with.

Before initiating the review of the Dynamic Links you must make sure you are working on the right Application by trying to match the analysis job name in the DLM tool against the Analysis Unit defined for the current version (see picture below):



Please check links you want to ignore: # Ignored links / # Links : 0/47

Calling type	Calling name	Calling type	Description
☐	com.salesmanager.central.orders.EditCustomerAction.editOrderEmailAddress	Java Method	Use
☐	com.salesmanager.central.ref.Types.load	Java Method	Use
☐	com.salesmanager.central.catalog.EditProductPriceDiscountAction.editProductPriceDiscount	Java Method	Use
☐	com.salesmanager.central.merchantstore.StoreFrontPortletsAction.preparePortletsPositions	Java Method	Use
☐	com.salesmanager.central.catalog.EditDiscountAction.saveDiscount	Java Method	Use
☐	com.salesmanager.central.catalog.EditImagesAction.deleteImage	Java Method	Use

Please use "Set as Link/NoLink" shortcut menu on method's hyperlink to parametrize:

Automated Dynamic Link review

Manually reviewing Dynamic Links (although a legitimate approach) is discouraged as it will not address the underlying cases that triggered the detection in the first place and it can be very time consuming, particularly if you have a large number of dynamic links to review. CAST therefore recommends the use of two options to automate this process:

- **Parametrization to automatically ignore or validate links** when they are created with a parameter of a method . Parameterization rules allow you to automatically exclude Dynamics Links when you re-run the analysis. Moreover, Parametrization rules can be reused in future analyses (via an Environment Profile) thus supporting the automation of the analysis process. CAST provides some default parametrization rules and you can also create your own.
- **Dynamic Link Manager rule files** - this option enables you to create XML based filter rules that can be tested via the command line and the applied in the CAST Management Studio GUI at Application and Analysis Unit or Technology level: each time an analysis is then run, the filter rules will be applied, either validating or ignoring links as required.

See the **CAST Management Studio help** for more information about both of the above options.

Technical notes

Limitation of the DLM source code viewer for C/C++

Viewing Source Code

A C++ link may possibly have several different associated pieces of code (in various files). When a server object is referenced in several files for a same Caller, CAST's Dynamic Link Manager will only display references found in one file.

For example:

```
<pre>file f1.h :
....
void f(const CString&amp; s = CString(&quot; T1&quot; ) );
...
file f2.cpp
...
void f( const CString&amp; s )
{
//...
LPSTR c = &quot;T1&quot;;
//...
}
...</pre>
```

In this example, T1 is a server object and function f references T1. The references are spread among two files, f1.h and f2.cpp.

Although there is more than one file that contains references to T1, the Dynamic Link Manager will only display one file (either f1.h or f2.cpp) and highlight all references in it. Thus all references in the other file will not be displayed. In this example, if the file displayed by Dynamic Link Manager is f2.cpp, there will be only one reference highlighted although there is another reference in f1.h. Therefore it can be difficult to decide if links to T1 are valid or should be ignored.

To workaround this problem, you can use the **Code Viewer in CAST Enlighten**. It displays all bookmarks. This allows dynamic links to be evaluated based on complete information.

Macros

Dynamic links will be created to macros based on the source code that is defined in a macro. However, when examining these dynamic links in the Dynamic Link Manager, the link will appear to originate in the macro and call any corresponding object based only on the strings that are defined in the macro. This is a functional limitation of the analyzer. In order to check the validity of these links, the corresponding file where the macro is defined will need to be opened manually.

Back to: 2.2.2. Validate and fine tune the Analysis