Automatic Links Validator - 1.0

Summary: This document provides information about the extension providing **automatic validation of dynamic links**.

Extension ID

com.castsoftware.automaticlinksvalidator

What's new ?

Please see **Automatic Links Validator - 1.0 - Release Notes** for more information.

Description

At the core of CAST AIP transaction discovery algorithm is the understanding of the links between objects discovered during the source code analysis of the target application. 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**" (see **Validate Dynamic Links** for more information. This extension provides **automatic validation** of these dynamic links.

In what situation should you install this extension?

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 **rejected** and removed from the Analysis Service. In many situations, however, **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. This extension is therefore aimed at situations where analysis results contain a very large number of dynamic links that need to be **validated automatically**.

When using AIP Console, this extension is installed automatically with every Application version that is analyzed. There is no need to manually install it.

What does it do?

On completion of an analysis, this extension will scan the results (stored in the Analysis Service schema) to **validate**, **reject** or **skip** the dynamic links automatically. The validation or rejection of a dynamic link is based on a series of heuristics which gave a score to the dynamic link:

- if > 0, the link is **validated as true**
- if < 0, the link is **rejected as false**
- if = 0, the link is **skipped** (generally this means that none of the heuristics can be applied to this link and in this case, you will need to **review the links manually** as explained in **Validate Dynamic Links**.

Note that:

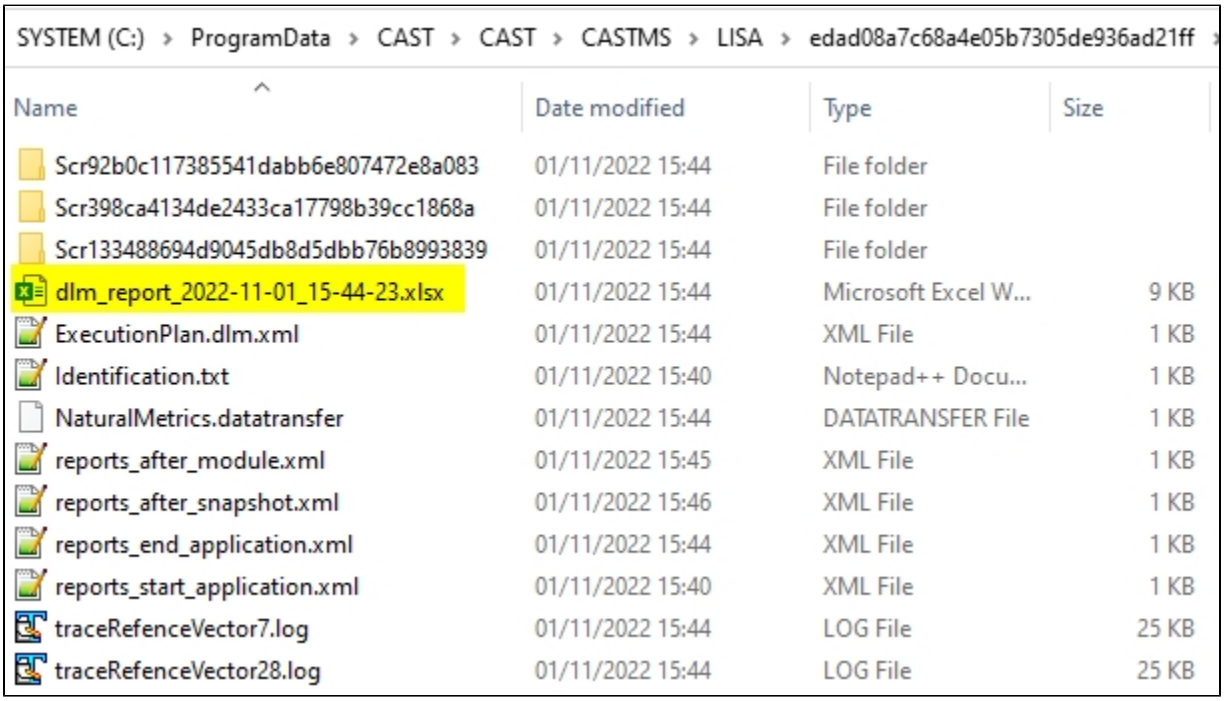
- only links that have not yet been manually reviewed or reviewed by this extension in a previous analysis will be pass through the validation process.
- links with several bookmarks are handled by the extension, the rule is: if at least one bookmark is validated, then the entire link is validated.
- the status of the link in the Analysis Service schema is modified following the validation process.

Report generation

Moreover a **Microsoft Excel report** is generated that contains:

- link information: caller, type, callee, and code of link
- the resulting action (validated, rejected, skipped)
- the description of the heuristics used

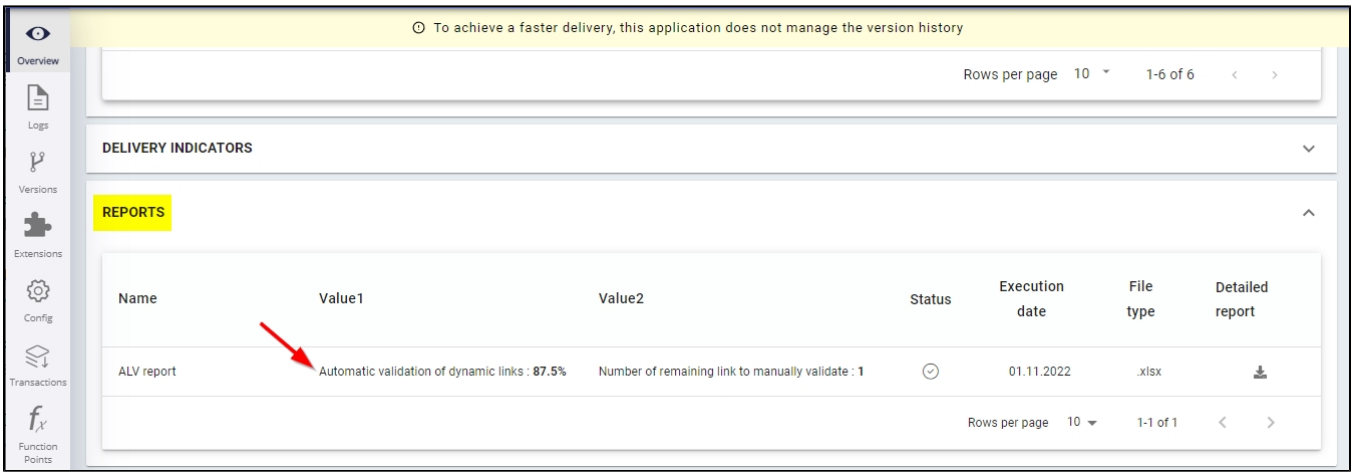
This Microsoft Excel report is stored in the **LISA** folder (Large Intermediate Storage Area) which is usually set to **%PROGRAMDATA%\CAST\CAST\CASTMS\LISA**:



Console 1.x

When using **Console 1.x**, the report can be accessed directly in **Application - Legacy Overview**:

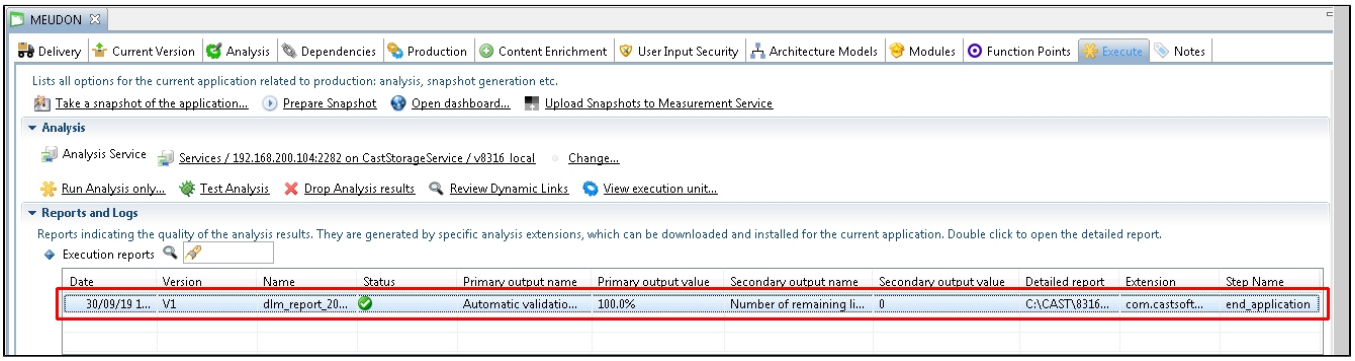
Click to enlarge

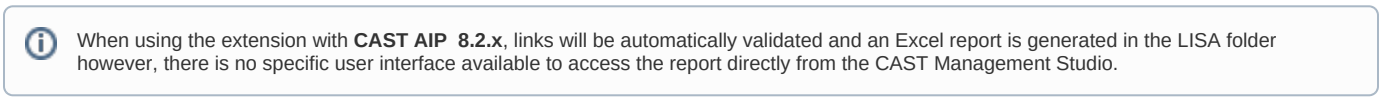


CAST Management Studio

The report is available in the CAST Management Studio (CAST AIP 8.3.x) in the **Execute tab** after performing an analysis:

Click to enlarge



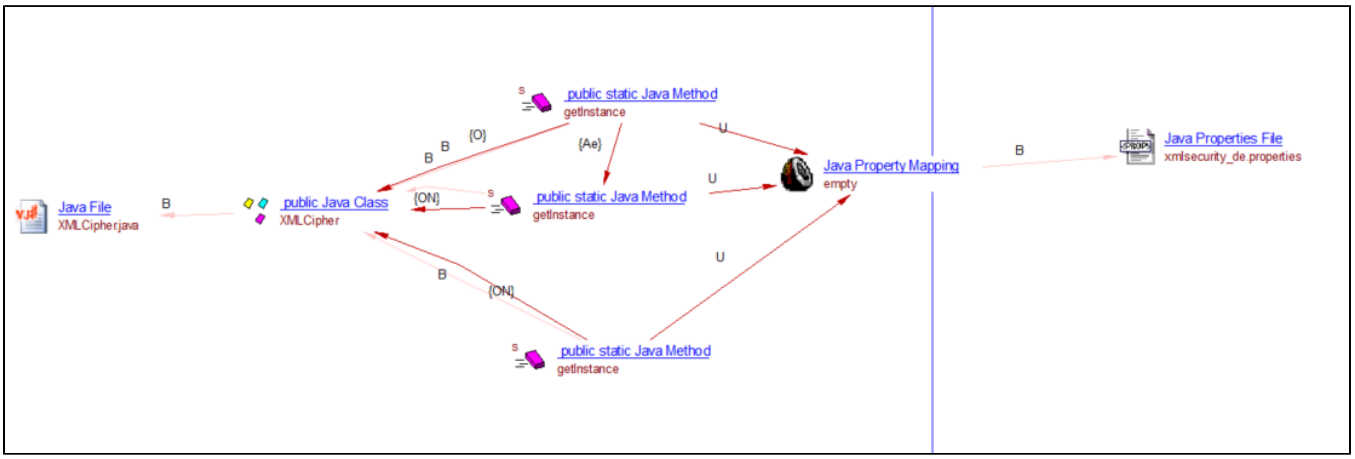


When using the extension with **CAST AIP 8.2.x**, links will be automatically validated and an Excel report is generated in the LISA folder however, there is no specific user interface available to access the report directly from the CAST Management Studio.

What results can you expect?

The vast majority of the dynamic links in the Analysis Service schema will be reviewed and either validated as true or rejected as false Below is an example of a view in CAST Enlighten, first **without the extension** and then **with the extension**. We see that three dynamic links have been (correctly) rejected as false:

Results without extension



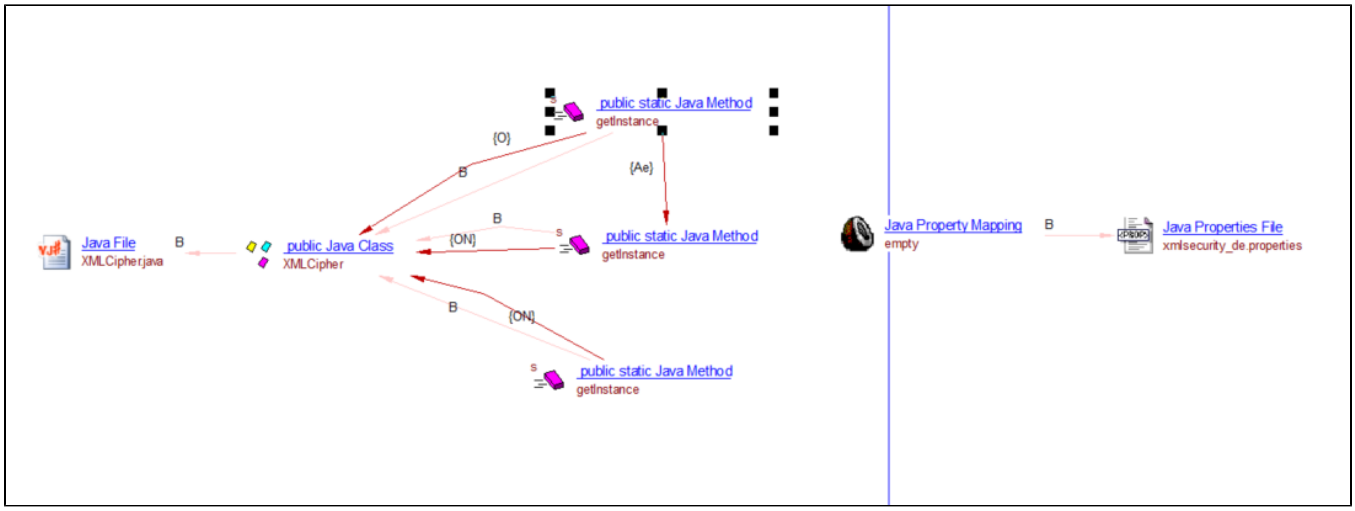
Below is the code of the first **'getInstance'** method: we can see that the reference is in a **throw exception**, so the link is not valid and needs to be rejected as false:

```
public static XMLCipher getInstance(String transformation, String canon)
      throws XMLEncryptionException
   {
      XMLCipher instance = XMLCipher.getInstance(transformation);

      if (canon != null)
      {
         try
         {
            instance._canon = Canonicalizer.getInstance(canon);
         }
         catch (InvalidCanonicalizerException ice)
         {
            throw new XMLEncryptionException("empty", ice);
         }
      }

      return instance;
   }
```

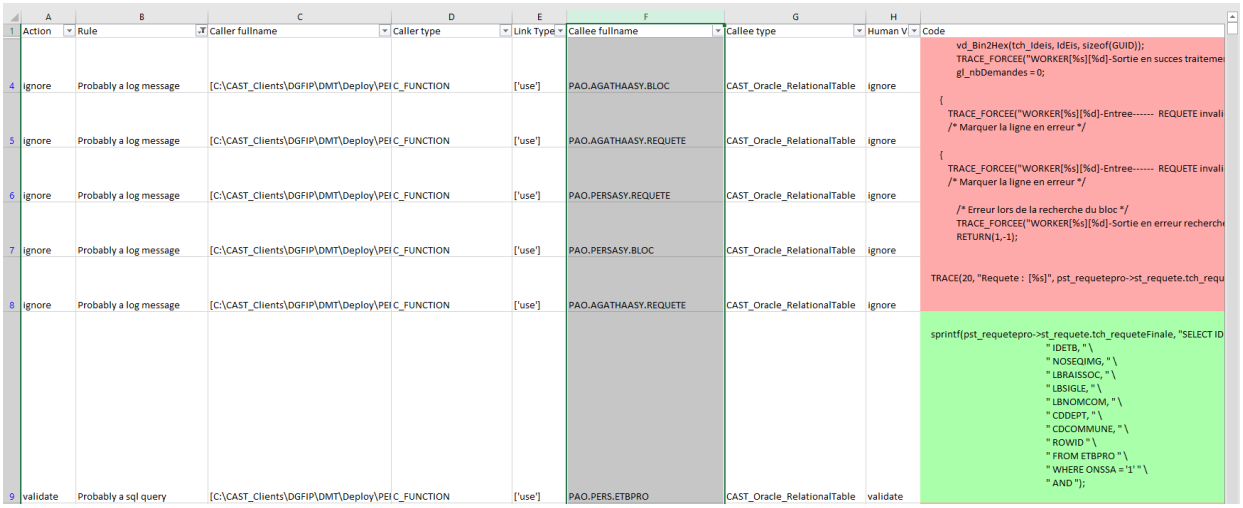
Results with extension - the false links have been rejected:



Report contents

Below an example of the Microsoft Excel report generated by the extension:

Click to enlarge



	A	B	C	D	E	F	G	H	
1	Action	Rule	Caller fullname	Caller type	Link Type	Callee fullname	Callee type	Human V	Code

4	ignore	Probably a log message	[C:\CAST_Clients\DGFIP\DMT\Deploy\PEI	C_FUNCTION	['use']	PAO.AGATHAASY.BLOC	CAST_Oracle_RelationalTable	ignore	vd_Bin2Hex(tch_Ideis, IdEis, sizeof(GUID));
									TRACE_FORCEE("WORKER[%s][%d]-Sortie en succes traiteme
									gl_nbDemandes = 0;

5	ignore	Probably a log message	[C:\CAST_Clients\DGFIP\DMT\Deploy\PEI	C_FUNCTION	['use']	PAO.AGATHAASY.REQUETE	CAST_Oracle_RelationalTable	ignore	{ TRACE_FORCEE("WORKER[%s][%d]-Entree------ REQUETE invali /* Marquer la ligne en erreur */
6	ignore	Probably a log message	[C:\CAST_Clients\DGFIP\DMT\Deploy\PEI	C_FUNCTION	['use']	PAO.PERSASY.REQUETE	CAST_Oracle_RelationalTable	ignore	{ TRACE_FORCEE("WORKER[%s][%d]-Entree------ REQUETE invali /* Marquer la ligne en erreur */
7	ignore	Probably a log message	[C:\CAST_Clients\DGFIP\DMT\Deploy\PEI	C_FUNCTION	['use']	PAO.PERSASY.BLOC	CAST_Oracle_RelationalTable	ignore	/* Erreur lors de la recherche du bloc */ TRACE_FORCEE("WORKER[%s][%d]-Sortie en erreur recherch RETURN(1,-1);
8	ignore	Probably a log message	[C:\CAST_Clients\DGFIP\DMT\Deploy\PEI	C_FUNCTION	['use']	PAO.AGATHAASY.REQUETE	CAST_Oracle_RelationalTable	ignore	TRACE(20, "Requete : [%s]", pst_requetepro->st_requete.tch_requ
9	validate	Probably a sql query	[C:\CAST_Clients\DGFIP\DMT\Deploy\PEI	C_FUNCTION	['use']	PAO.PERS.ETBPRO	CAST_Oracle_RelationalTable	validate	sprintf(pst_requetepro->st_requete.tch_requeteFinale, "SELECT ID " IDETB, " \ " NOSEQIMG, " \ " LBRAISSOC, " \ " LBSIGLE, " \ " LBNOMCOM, " \ " CDDEPT, " \ " CDCOMMUNE, " \ " ROWID " \ " FROM ETBPRO " \ " WHERE ONSSA = '1' " \ " AND ");

The report contains several sheets/tabs:

- **Automatic DLM**: This sheet shows all the link information, corresponding actions and descriptions of the heuristics used
- **Remaining links**: This sheet shows the links which haven't been successfully validated or rejected and in this case, you will need to **review the links manually** as explained in **Validate Dynamic Links**.
- **Summary**: This sheet show numbers summarizing the results of the process
    1. Number of dynamic links
    2. Number of links handled by the extension
    3. Number of links validated, ignored or skipped
    4. Rates of handling, validating, ignoring or skipping links
- **Conflicting links**: Links assessed with conflicts. The links have been checked with clear results but with both validating and ignoring rules. These links will need to be **reviewed manually** as explained in **Validate Dynamic Links** as they are more likely to have an incorrect assessment

AIP Core compatibility

This extension is compatible with:

AIP Core release	Supported
8.0.0	✅

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.

> Note that when using AIP Console, this extension is automatically installed when:
>
> - using **Workflow - Modern application onboarding**
> - using **Workflow - Legacy application onboarding** and when you choose an **Objective** - see **Legacy application onboarding - Standard onboarding - perform all actions - choose objectives**

How does it work?

Mechanics of the validation process

1. The extension checks the dynamic link against a series of heuristics
2. Each heuristic gives a score (positive or negative) to each dynamic link
3. All scores are added up to give a final score = .
4. The decision to validate as true, reject as false or skip the links is based on the value of :

- if > 0, the link is **validated as true**
- if < 0, the link is **rejected as false**
- if = 0, the link is **skipped** (generally this means that none of the heuristics can be applied to this link and in this case, you will need to **review the links manually** as explained in **Validate Dynamic Links**)

5. In AIP Core 8.3.x a Microsoft Excel report is generated and stored in the LISA folder (Large Intermediate Storage Area) containing information about the status of each link after validation

General information about dynamic links

The application source code is parsed and investigated by analyzers. From this analysis, references to other objects are detected and links are created when appropriate. These links are tagged as "**dynamic**". Not all links generated in this fashion are valid and their **validation** by the CAST AI Admin is therefore **required**.

Note that the term 'dynamic' is ambiguous: calling them 'grep' would be more in accordance with the reality. These links are also described as 'not sure' compared to links created by parsing associated with resolution. As a consequence they need "validation". The most common example of such links concerns parsed strings:

```
std::string message = "SELECT * FROM table";
```

One of the main objectives of the presence of dynamic links is to be able to see links to a database, even when the SQL code is in client code strings and in unsupported frameworks.

Primary heuristic

Inside a program, a string may either be:

- a string that will be interpreted by a human: log, message, ui etc.
- a string that represents another code, or part of a code to be interpreted by a program : SQL, name of resource etc...

In the first case the dynamic link is incorrect. In the second case, it is 'correct'; at least in the sense of 'grep'.

Description of the heuristics used by the extension

Heuristic	Rationale
Ignore throws exception	String in a 'throw' exception is always a message to be interpreted by a human, so the link is invalid.
Skip reference finder	Reference finder link, the extension will skip them and not process any heuristic on it.
Ignore message logging	Log messages are to be interpreted by human, so the link is invalid.
Ignore SQL parameter	SQL parameter are not valid link.
Ignore WPF property changed	Reference is RaisePropertyChanged(\"ObjectName\"), this is a classic WPF construct, so an invalid link.
Validate or ignore when the Reference is a path	Validate a reference which is valid path file and the callee object is a file.
Validate call to program	_
Validate or ignore link to properties element	Validate or ignore link to JSP property
Validate or ignore SQL query	Validate correct SQL query syntax
Validate C# call procedure	Known functions call to database procedure**.**
Validate link to Spring bean	_
Ignore link JSP servlet mapping	Ignore link to JSP servlet mapping.
Ignore link from properties element to properties element	Ignore link from JSP property to JSP property.
Ignore properties element when it's a message logging	Ignore link when caller is a JSP_PROPERTY_MAPPING and its name contains a log marker
Ignore link to natural language	Ignore link when the reference is in a string of natural language.
Ignore link to directory	Ignore link to a directory (a directory is not an end point neither can it calls a link).
Ignore link to a column of a table	Ignore link to a column table (the link should be to a table).
Ignore link to synonym	Ignore link to a synonym (the link should be to a table).

Ignore link to a wrong type of callee	Ignore link to a wrong type of callee.
Validate .NET DataTable links	Validate link using method from ADO .Net DataTable.
Ignore link when the caller is a sourceFile	Ignore link when caller is a sourceFile and the callee is not a sourceFile.
Ignore link on database index	Ignore link when callee is an index of a database.
Validate .NET ObjectContext methods	Validate link using method from .Net ObjectContext.
Validate link to JPA Persistence XML	Validate link to JPA Persistence XML file.
Ignore link from JSP file to table	Ignore link from JSP file to table with callee in a tag.
Validate link to SEARCHSTRING	Validate link to a REFIND_SEARCHSTRING callee.
Validate link to a java applet	_
Validate link from .NET object to ENTITY_WRAPPER object	_
Ignore invalid Struts or Spring links	Ignore Struts or Spring links with wrong type of callee (it's a common DLM rule).
Validate or Ignore link from Java field to JPA	Validate link with "JV_FIELD" caller and "JPA_NAMED_QUERY" callee when the field is strictly equal to jpa_entity.jpa_named_query. Ignore link with "JV_FIELD" caller and "JPA_ENTITY" callee when the field is strictly equal to jpa_entity.jpa_named_query.
Ignore link from toString methods	_
Validate or Ignore link to JspForward	Ignore link to callee of type JSP_FORWARD unless a part of the fullname is found in the source.
Ignore link from wrong method or function	Some standard methods or functions can't be used to call an object (typically manipulation of string, etc.)
Ignore link to wrong type of callee object from another technology	Some object type can't be called from "outside" their technology
Ignore link with pattern callee_name.XXX or callee_nameXXX in code	_
Ignore link when caller is an exception handler	_
Ignore link when caller is exception constructor	_

Conflicting links

As mentioned already, each heuristic rule computes a score for each link which can be positive or negative. A positive score will weight in favor of **validating the link** and a negative score in favor of **rejecting it**. A conflicting exists when a link obtains **positive and negative scores** regardless of the value of the final score. These links are worth mentioning because they are the ones where there is the highest risk of an incorrect assessment.

In the case of these links the difficulty lies in the estimation of the marks for each rule and some times a choice has been that deserves an explanation:

Validating rule	Ignoring rule	Results	Rationale
Link to properties element as argument of a call	Probably a log message	IGNORE	A log message is a dead end in transaction analysis and here we typically have an insertion of the property value in a log message. Yes, the link toward the "property" is real but as it is inserted in a log message it has no real value as a link. Moreover if we validate this link we create a risk: if an incorrect analysis is done of the content of the "property" then there is a risk of creating a false transaction. We choose the safe choice of ignoring these links which has more value in the global analysis of an application.
Link to properties element	Probably a log message	IGNORE	A log message is a dead end in transaction analysis and here we typically have an insertion of the property value in a log message. Yes, the link toward the "property" is real but as it is inserted in a log message it has no real value as a link. Moreover if we validate this link we create a risk : if an incorrect analysis is done of the content of the "property" then there is a risk of creating a false transaction. We choose the safe choice of ignoring these links which has more value in the global analysis of an application.
Link to properties element as argument of a call	This is probably natural language	IGNORE	Natural language is destined to be read by human, it is a dead end in transaction analysis and here we typically have an insertion of the property value in this message. Yes, the link toward the "property" is real but as it is inserted in a natural language message it has no real value as a link. Moreover if we validate this link we create a risk: if an incorrect analysis is done of the content of the "property" then there is a risk of creating a false transaction. We choose the safe choice of ignoring these links which has more value in the global analysis of an application.

Link to properties element	This is probably natural language	IGNORE	Natural language is destined to be read by human, it is a dead end in transaction analysis and here we typically have an insertion of the property value in this message. Yes, the link toward the "property" is real but as it is inserted in a natural language message it has no real value as a link. Moreover if we validate this link we create a risk : if an incorrect analysis is done of the content of the "property" then there is a risk of creating a false transaction. We choose the safe choice of ignoring these links which has more value in the global analysis of an application.
Link to properties element as argument of a call	This is a throw exception, so an invalid link	IGNORE	An exception message is a dead end in transaction analysis and here we typically have an insertion of the property value in an exception message. Yes, the link toward the "property" is real but as it is inserted in an exception message it has no real value as a link. Moreover if we validate this link we create a risk : if an incorrect wrong analysis is done of the content of the "property" then there is a risk of creating a false transaction. We choose the safe choice of ignoring these links which has more value in the global analysis of an application.
Link to properties element	This is a throw exception, so an invalid link	IGNORE	An exception message is a dead end in transaction analysis and here we typically have an insertion of the property value in an exception message. Yes, the link toward the "property" is real but as it is inserted in an exception message it has no real value as a link. Moreover if we validate this link we create a risk : if an incorrect analysis is done of the content of the "property" then there is a risk of creating a false transaction. We choose the safe choice of ignoring these links which has more value in the global analysis of an application.

Run the extension independently

This section present the method to run the extension independently of an analysis and directly on a knowledge base. It can be useful if you have already performed your analysis without having installed the extension or if you want to use a new version of the extension on an old analysis.

> **Info**
>
> **The extension can be run independently only if the application has already been analyzed.**

You have two possibilities:

- run directly the extension via a python script with a list of arguments;
- run the batch script run.bat present in the extension.

> **Warning**
>
> It is strongly advised to use the python interpreter of your version of CAIP. If not you take the risk of missing libraries (cast extension SDK for example).
>
> The interpreter can be found in the folder "ThirdParty\Python34" of CAIP.

Using a python command line to run the extension

The script is located in main.py file of the extension folder. The command is the following:

```
/path to python interpreter/python /path to com.castsoftware.automaticlinksvalidator/main.py cmd kb_name
application_name src_code_root_path  [-l LOCAL_SRC_ROOT_PATH] [-r REPORT_PATH] [-p REPORT_PREFIX] [-n] [-a] [-d]
```

Where:

- **cmd** asks for the command line run MANDATORY;
- **kb_name** is the name of the knowledge base used for the analysis MANDATORY;
- **application_name** is the name of the application MANDATORY;
- **src_code_root_path** is the path to the root folder of the code used for the analysis MANDATORY;
- **local_scr_root_path** is the path to the root folder of the code if it is not the same used for the analysis (only interesting if you have retrieved a kb and the source code of the application);
- **report_path** path to the folder where you want the report to be put;
- **report_prefix** is the prefix for the report, by default it is the application name;
- **-n** specifies that you do not want the extension to modify the knowledge base (useful if you're only interested in the report);
- **-a** specifies that you want the extension to check **all** dynamic links including those which are already validated or ignored (To be use with strong caution as it will probably changes results);
- **-d** specifies that you want the development report (only useful for developers of the extension);

Using the script run.bat

Fills the mandatory fields and the optional parameters in the script and run it.

- **aip_path** is the path to AIP MANDATORY;
- **automaticlinksvalidator_path** is the path to the extension automaticlinksvalidator MANDATORY;

- **kb_name** is the name of the knowledge base used for the analysis MANDATORY;
- **application_name** is the name of the application MANDATORY;
- **kb_src_root_path** is the path to the root folder of the code used for the analysis MANDATORY;
- **local_src_root_path** is the path to the root folder of the code if it is not the same used for the analysis (only interesting if you have retrieved a kb and the source code of the application);
- **report_path** is the path to the folder where you want the report to be put;
- **report_prefix** is the prefix for the report, by default it is the application name;
- **not_apply_validation** specifies that you do not want the extension to modify the knowledge base (useful if you're only interested in the report);
- **review_all_dynamic_links** specifies that you want the extension to check **all** dynamic links including those which are already validated or ignored (To be use with **strong caution** as it will probably changes results);
- **development_report** specifies that you want the development report (only useful for developers of the extension);