Application consistency review

This documentation is no longer maintained and may contain obsolete information. You should instead refer to **Application onboarding**.

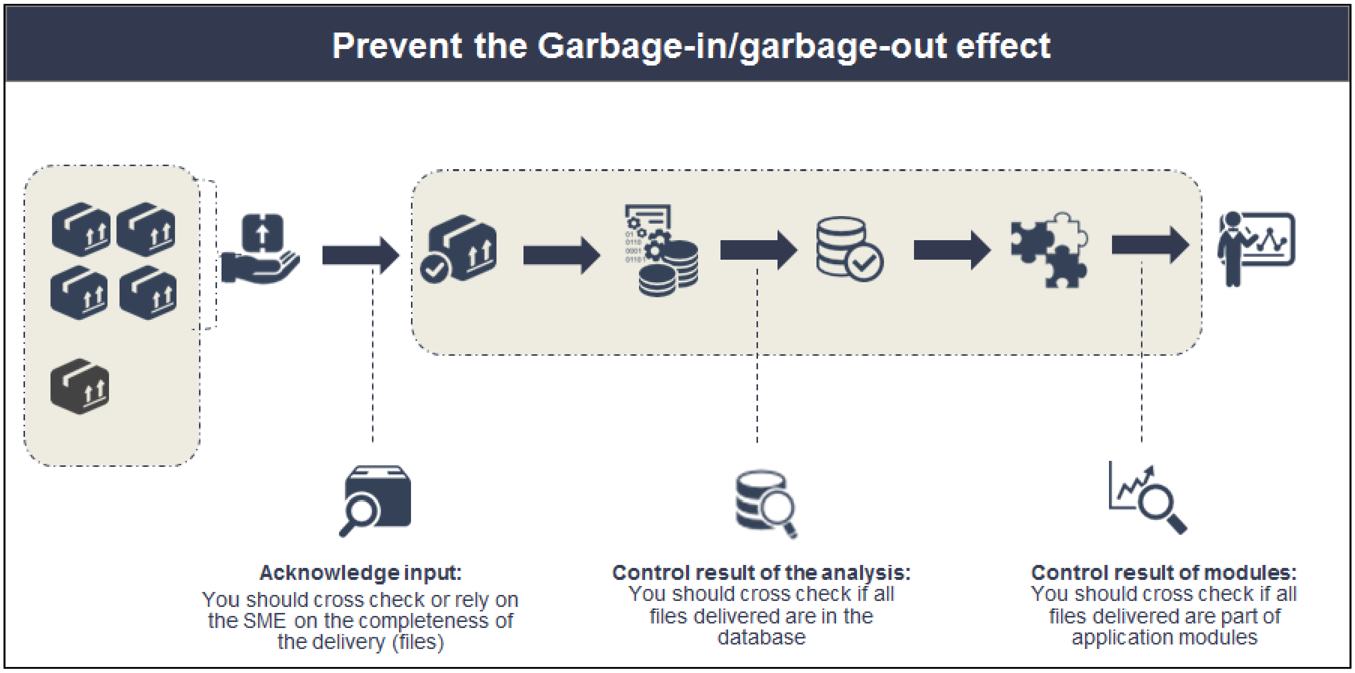
On this page:

Introduction

Two aspects are taken in to account when measuring the software functional size: the application boundary and the changes done on features throughout the application life cycle. It is extremely important to ensure that the measurement process is as accurate as possible and then to control the measurement process and to validate source code, the application boundary, the configuration, and finally the results.

Control of the measurement process can be achieved through four steps:

- The first one is to compare the source code delivery with the source code selection in analysis settings. If necessary, adjustments can be done in the CAST Delivery Manager Tool (DMT) package settings by the AI Administrator.
- The next step is to compare the source code that has been analyzed with the resulting objects in the Analysis Service. Customization can be implemented in the CAST Management Studio to take in to account specific technologies or coding.
- The third step involves checking the application module content. Modules are used to compute indicators for specific parts of the application and, as a consequence, it is important to justify all the exclusions of objects.
- The last step is to review the list of empty transactions with regard to the configuration that has been set up by the AI Administrator. Empty transactions can be considered as excluded from the measurement results and, as such, they must be validated.



Prevent the Garbage-in/garbage-out effect
Acknowledge input:
You should cross check or rely on the SME on the completeness of the delivery (files)
Control result of the analysis:
You should cross check if all files delivered are in the database
Control result of modules:
You should cross check if all files delivered are part of application modules

Check the application boundary consistency

Because the application boundary is the aggregation of all the components that compose the application, all the source code that has been provided to the AI Center should be considered. As a consequence, it is critical to compare the same code that has been delivered with the code that is visible in the various CAST dashboards.

Based on the architecture review you should have a good overview of the different technologies and interfaces used by the application to communicate with other applications and end-users. Several mechanisms are available to adjust the application boundary throughout the on-boarding process with specific coding requiring additional pieces of code that are not necessarily maintained by the application team.

The application boundary can be refined in several steps:

a. **Check the source code delivery in the CAST Delivery Manager Tool (DMT):** The source code that is used as input for the application analysis should be the one you consider as the application boundary. In this case all the source code delivered through the DMT packages should be in the Analysis Service and part of the application transactions.
b. **Adjust analysis settings in the CAST Management Studio (CMS):** If the source code perimeter has not been refined in DMT packages, then you can also fine tune the corresponding Analysis Units created in CMS.
c. **Tune the module content:** If you still have to keep some elements during the application analysis but you do not want to consider them in the results, then you should refine the module definition in CMS.
d. **Configure transaction in Transaction Configuration Center (TCC):** Finally, if you have to keep some modules of the application to display quality measures and you do not want to consider them in the functional sizing, then exclusions in the CAST Transaction Configuration Center (TCC) can be an option to consider.

Consistency between DMT package input and DMT package output

It is critical to ensure that all delivered source code will be "consumed" by CAST AIP. Pieces of source code that cannot be part of the application analysis must be documented as being excluded from the application boundary. Get the list of files that contain the application source code by executing the following Windows batch command:

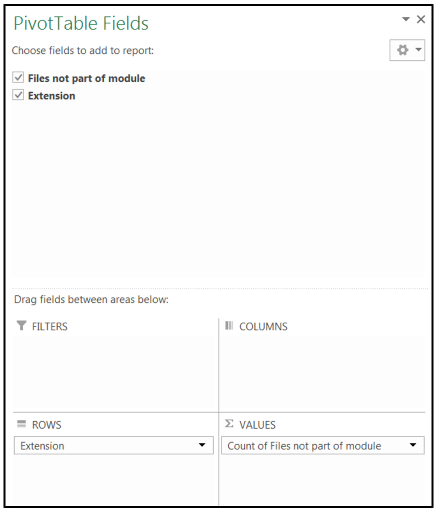
```
dir /a:a/s/b/o:n >listFiles.csv
```

Open the resulting CSV file with a tool like Excel. You will see in column A one row per file. Apply the following formula to the column B in order to get the actual list of file extensions:

```
=RIGHT($A2,LEN($A2)-FIND("|",SUBSTITUTE($A2,".","|",LEN($A2)-LEN(SUBSTITUTE($A2,".","")))))
```

Select the worksheet, and create a pivot table with the extensions as rows and corresponding number of files as value:

Click to enlarge:



PivotTable Fields

Choose fields to add to report:

☑ Files not part of module
☑ Extension

Drag fields between areas below:

FILTERS

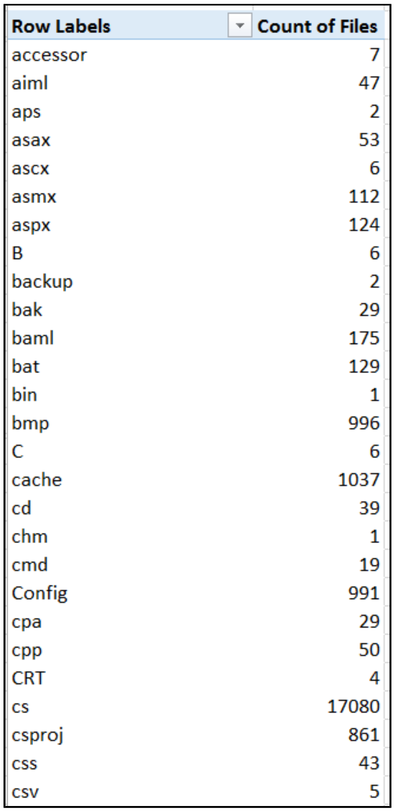
COLUMNS

ROWS: Extension

VALUES: Count of Files not part of module

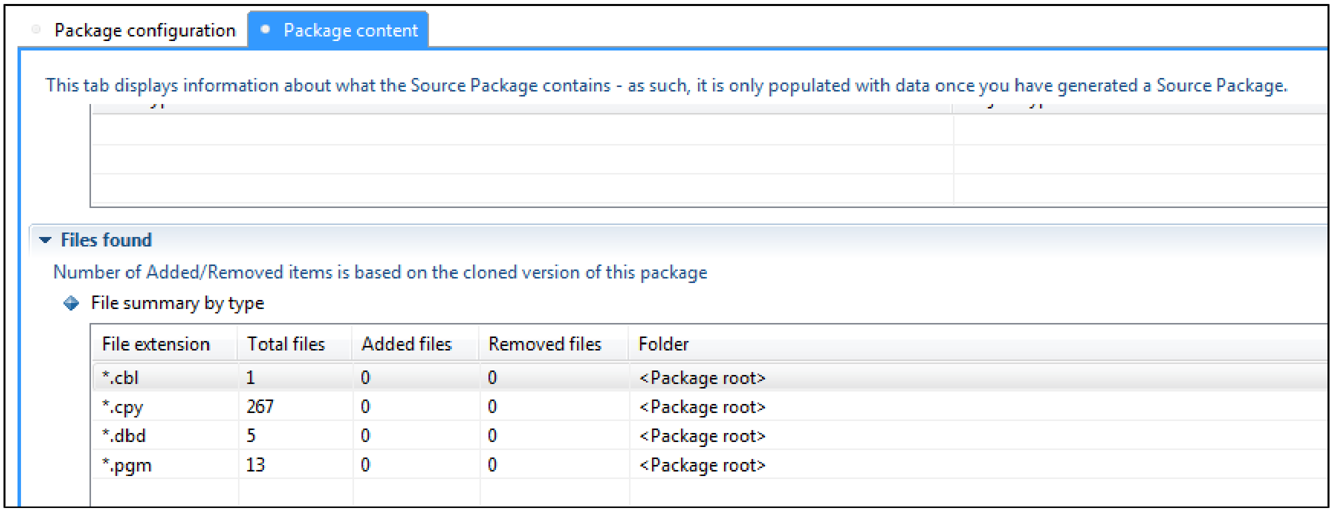
The following image displays the resulting pivot table:

Click to enlarge:



Row Labels	Count of Files
accessor	7
aiml	47
aps	2
asax	53
ascx	6
asmx	112
aspx	124
B	6
backup	2
bak	29
baml	175
bat	129
bin	1
bmp	996
C	6
cache	1037
cd	39
chm	1
cmd	19
Config	991
cpa	29
cpp	50
CRT	4
cs	17080
csproj	861
css	43
csv	5

You can then search for missing files by comparing this list with the actual list displayed in the CAST Delivery Manager Tool "Package content" tab:



This tab displays information about what the Source Package contains - as such, it is only populated with data once you have generated a Source Package.

Files found

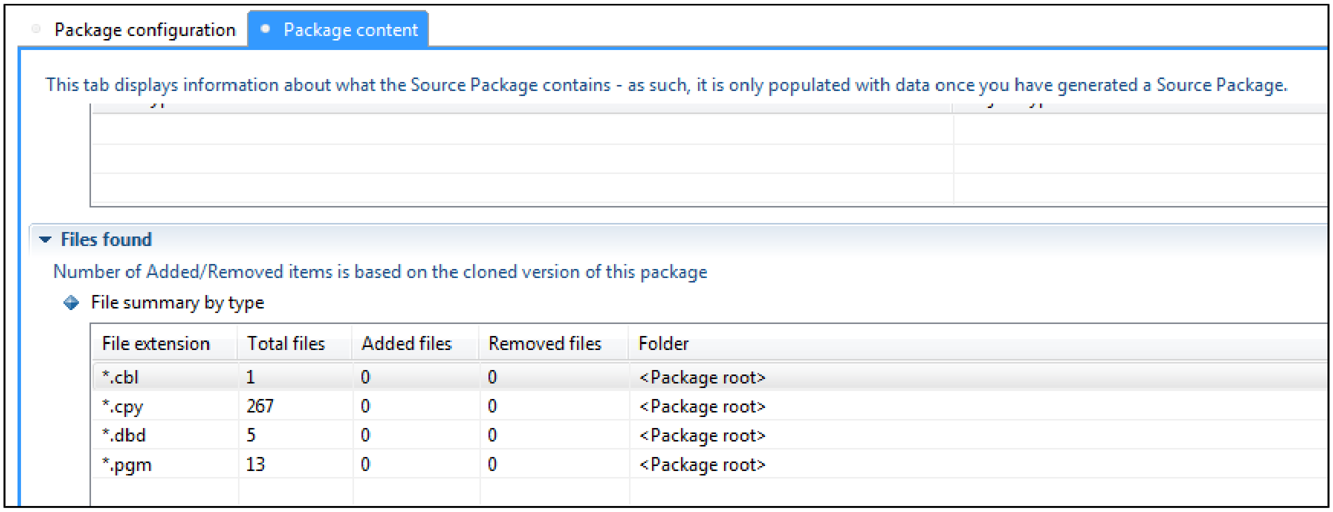
Number of Added/Removed items is based on the cloned version of this package

File summary by type

File extension	Total files	Added files	Removed files	Folder
*.cbl	1	0	0	<Package root>
*.cpy	267	0	0	<Package root>
*.dbd	5	0	0	<Package root>
*.pgm	13	0	0	<Package root>

Consistency between application analysis input and application analysis output

The CAST Delivery Manager Tool package content will be the input for the application analysis. The CAST Delivery Manager Tool "Package content" tab displays the various file extensions that have been discovered:



This tab displays information about what the Source Package contains - as such, it is only populated with data once you have generated a Source Package.

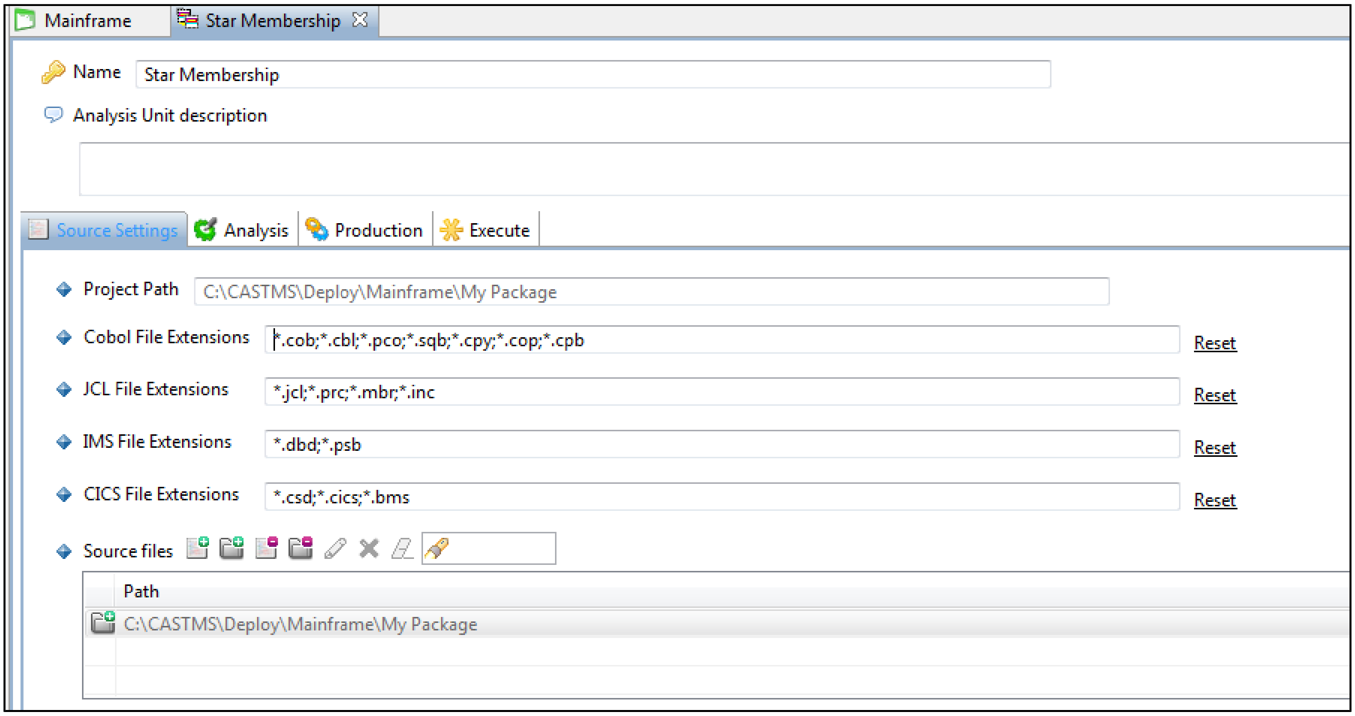
Files found

Number of Added/Removed items is based on the cloned version of this package

File summary by type

File extension	Total files	Added files	Removed files	Folder
*.cbl	1	0	0	<Package root>
*.cpy	267	0	0	<Package root>
*.dbd	5	0	0	<Package root>
*.pgm	13	0	0	<Package root>

You can compare this list with the file extensions that are specified in the CAST Management Studio in the Analysis Unit "Source Settings" tab. In the following example, the file extension "PGM" has been discovered in the CAST Delivery Management Tool package but is not present in the Analysis Unit "Source Settings" file extension list. It means that only the COBOL programs with the file extension "CBL" will be analyzed and those with the "PGM" file extension (13 programs) will not. To take in to account these files, you must add the missing file extensions in the Analysis Unit "Source Settings":



Name: Star Membership

Analysis Unit description

Project Path: C:\CASTMS\Deploy\Mainframe\My Package

Cobol File Extensions: *.cob;*.cbl;*.pco;*.sqb;*.cpy;*.cop;*.cpb

JCL File Extensions: *.jcl;*.prc;*.mbr;*.inc

IMS File Extensions: *.dbd;*.psb

CICS File Extensions: *.csd;*.cics;*.bms

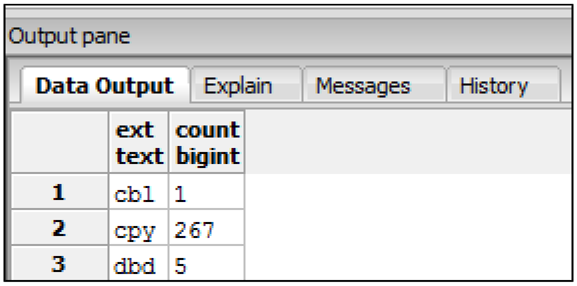
Source files

Path
C:\CASTMS\Deploy\Mainframe\My Package

Once the file extensions are coherent in both the CAST Delivery Manager Tool package and the CAST Management Studio Analysis Unit, then you should now expect to get the same number of files in the Analysis Service schema after the analysis has completed. To check that you can query the Analysis Service to get the number of source files that have been analyzed and compare them to the numbers displayed in the CAST Delivery Manager Tool "Package Content" tab. you should execute the following SQL query against the Analysis Service schema to get the number of files for the file extensions that have been taken in to account during the analysis:

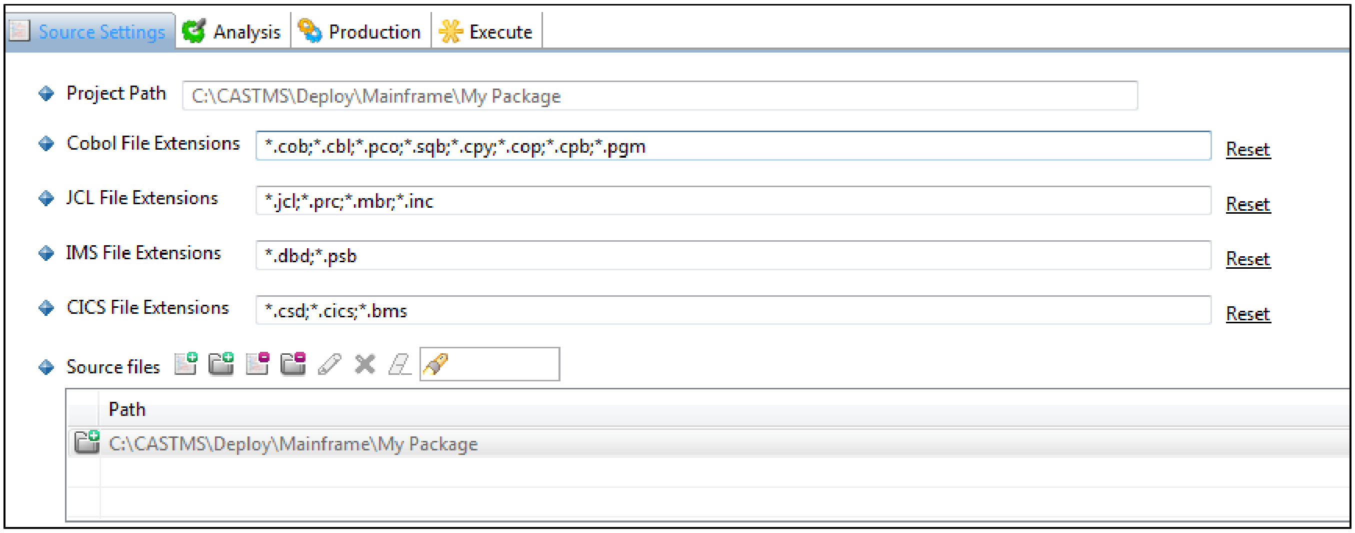
```
set search_path=<prefix>_local;
select substring(p.path from '\.([a-z]+)$') ext, count(*)
from (select distinct path from RefPath) p
group by ext;
```

The figure below displays the results returned by the query. They show that the files with the extension "PGM" have not been taken in to account during the application analysis:



	ext text	count bigint
1	cbl	1
2	cpy	267
3	dbd	5

This can be fixed by adding the "PGM" file extension in the Analysis Unit "Source Settings" tab:



Project Path: C:\CASTMS\Deploy\Mainframe\My Package

Cobol File Extensions: *.cob;*.cbl;*.pco;*.sqb;*.cpy;*.cop;*.cpb;*.pgm

JCL File Extensions: *.jcl;*.prc;*.mbr;*.inc

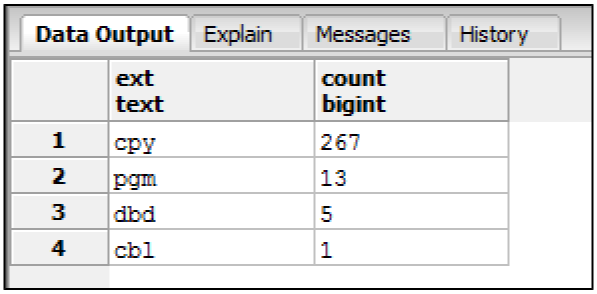
IMS File Extensions: *.dbd;*.psb

CICS File Extensions: *.csd;*.cics;*.bms

Source files

Path
C:\CASTMS\Deploy\Mainframe\My Package

Analyze the application again and then execute the SQL query. You should see the expected "PGM" files in the result set:



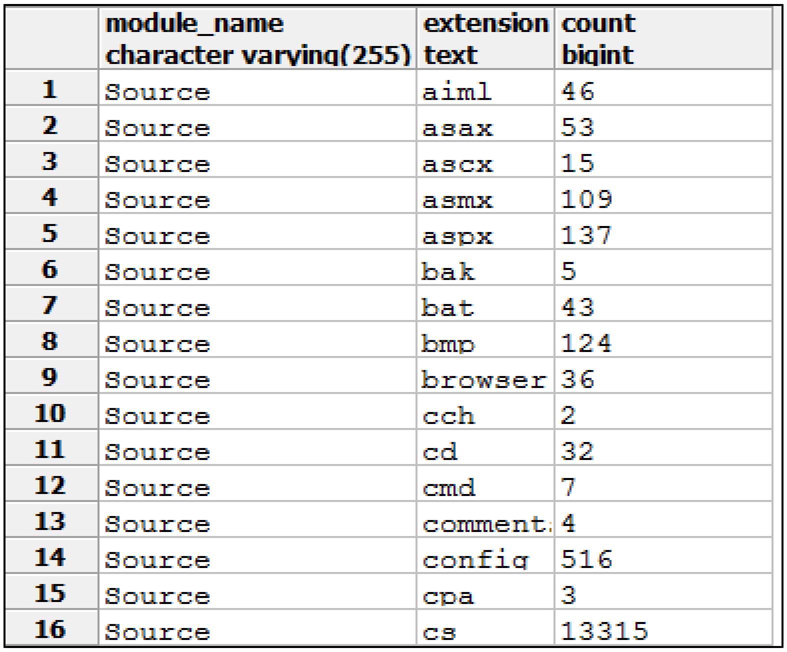
	ext text	count bigint
1	cpy	267
2	pgm	13
3	dbd	5
4	cbl	1

Consistency between application analysis output and module content

Use the following SQL query to check the content of the modules associated to the application:

```
set search_path=<prefix>_local;

SELECT ps.MODULE_NAME, SUBSTRING(cob.OBJECT_FULLNAME FROM '\.([a-z]+)$') as extension, count(*)
FROM <prefix_local>.PMC_SUBSET_OBJECTS pso
 JOIN (SELECT ps.SUBSET_ID as MODULE_ID,
 pm.OBJECT_NAME as MODULE_NAME
 FROM (SELECT pm.OBJECT_ID,
 pm.OBJECT_NAME
 FROM <prefix_mngt>.CMS_PORTF_MODULE pm) pm
 JOIN <prefix.local>.PMC_SUBSETS ps
 ON ps.SUBSET_NAME LIKE 'CMS_MOD__' || pm.OBJECT_ID || '_Preparation2'
 ) ps
 ON pso.SUBSET_ID = ps.MODULE_ID
 JOIN <prefix_local>.CDT_OBJECTS cob
 ON cob.object_id = pso.object_id
 AND cob.OBJECT_TYPE_STR LIKE '%File'
GROUP BY MODULE_NAME, extension, cob.OBJECT_TYPE_STR
ORDER BY 1 ASC, 2 ASC;
```

The result should look like this:

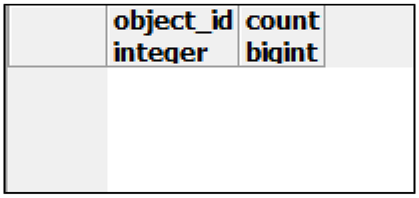


	module_name	extension	count
	character varying(255)	text	bigint
1	Source	aiml	46
2	Source	asax	53
3	Source	ascx	15
4	Source	asmx	109
5	Source	aspx	137
6	Source	bak	5
7	Source	bat	43
8	Source	bmp	124
9	Source	browser	36
10	Source	cch	2
11	Source	cd	32
12	Source	cmd	7
13	Source	comment.	4
14	Source	config	516
15	Source	cpa	3
16	Source	cs	13315

It is important to be sure a file has not been assigned to **multiple modules**. This can be checked by executing the following SQL query against the Analysis Service schema:

```
set search_path=<prefix>_local;

SELECT pso.OBJECT_ID, count(*)
FROM <prefix_local>.PMC_SUBSET_OBJECTS pso
JOIN (SELECT ps.SUBSET_ID as MODULE_ID,
pm.OBJECT_NAME as MODULE_NAME
FROM (SELECT pm.OBJECT_ID_ID,
pm.OBJECT_NAME
FROM <prefix_mngt>.CMS_PORTF_MODULE pm ) pm
JOIN <prefix_local>.PMC_SUBSETS ps
on ps.SUBSET_NAME like 'CMS_MOD__' || pm.OBJECT_ID || '_Preparation2'
) ps
on pso.SUBSET_ID = ps.MODULE_ID
GROUP BY OBJECT_ID
HAVING count(*) >1;
```

The result should be empty:



	object_id integer	count bigint

Consistency between module content and transaction graph

Once you reach this step, you are sure the input for the transaction building engine is (almost) correct. The resulting transaction call graphs must now be reviewed and validated. Several points should be checked and are presented below.

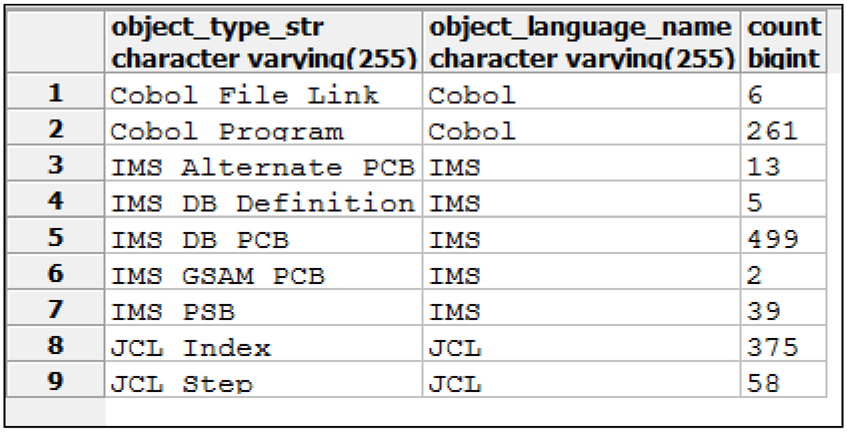
> Before investigating the transactions that have been discovered by CAST AIP, you may have to adjust the list of Transaction Entry Points - see Transaction configuration.

Objects that contribute to a transaction

The following SQL query will extract the object types that are part of a transaction. This is illustrated with COBOL in the following example but the query can be adapted to other object types:

```
set search_path=<prefix>_local;
select object_type_str,object_language_name,count(1)
from CDT_OBJECTS where object_id not in (
    select distinct objc.object_id
    from dss_transaction dt, dss_transactiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
    where dt.form_id =  obj.object_id
    and objc.object_id = dtd.child_id
    and dt.object_id = dtd.object_id
  union all
    select distinct objc.object_id
    from dss_datafunction dt, dss_datafunctiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
    where dt.maintable_id =  obj.object_id
    and objc.object_id = dtd.table_id
    and dt.object_id = dtd.object_id
)
and object_fullname not like '[Unknown%'
and object_language_name != '<N/A>'
and object_language_name != 'N/A'
and object_type_str not like  '%Project'
and object_type_str not like  '%Directory'
and object_type_str not like  '%Folder'
and object_type_str not in ('Cobol Paragraph','Cobol Section','Cobol CopyBook','Cobol Data Link','Cobol Entry
Point')
 group by 1,2
 order by 2,1;
```

The result should look like this:



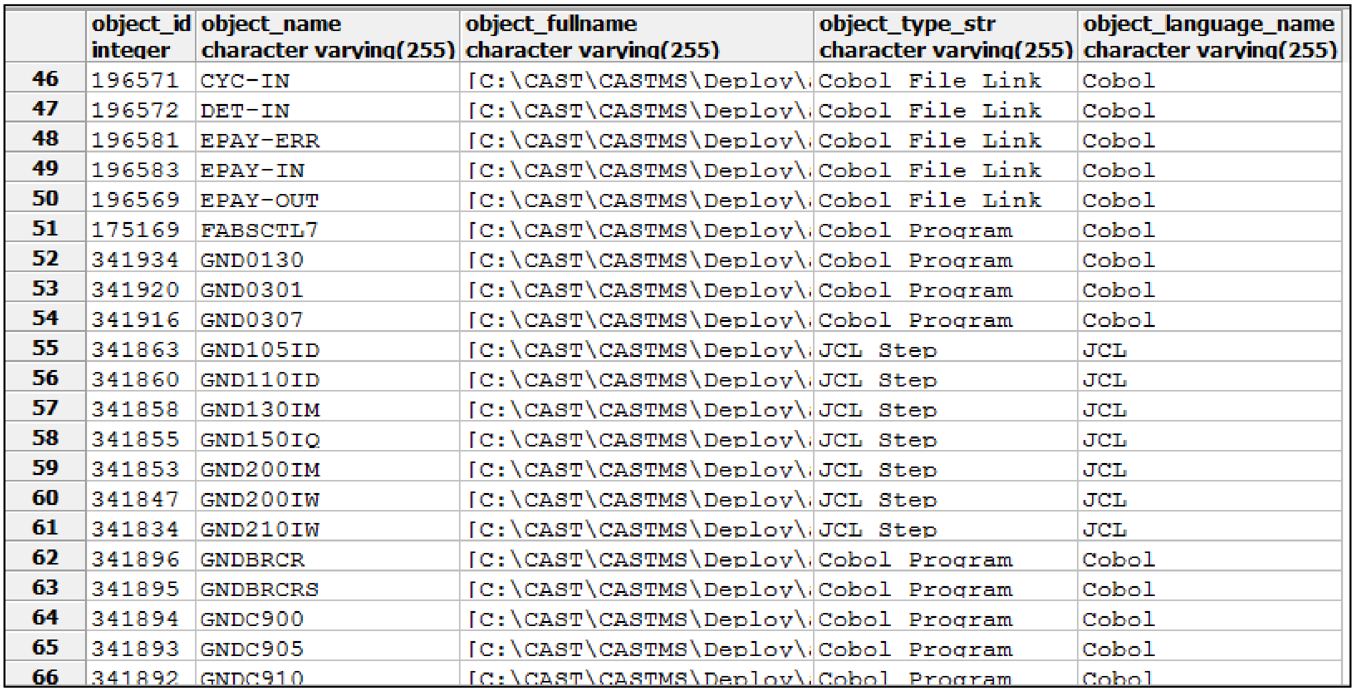
	object_type_str character varying(255)	object_language_name character varying(255)	count bigint
1	Cobol File Link	Cobol	6
2	Cobol Program	Cobol	261
3	IMS Alternate PCB	IMS	13
4	IMS DB Definition	IMS	5
5	IMS DB PCB	IMS	499
6	IMS GSAM PCB	IMS	2
7	IMS PSB	IMS	39
8	JCL Index	JCL	375
9	JCL Step	JCL	58

Objects that do not contribute to any transaction

The following SQL query will extract the object types that are not in any transaction. This is illustrated with COBOL in the following example but the query can be adapted to other object types:

```
set search_path=<prefix>_local;

select object_id, object_name, object_fullname, object_type_str,object_language_name
from CDT_OBJECTS where object_id not in (
    select distinct objc.object_id
    from dss_transaction dt, dss_transactiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
    where dt.form_id =  obj.object_id
    and objc.object_id = dtd.child_id
    and dt.object_id = dtd.object_id
  union all
    select distinct objc.object_id
    from dss_datafunction dt, dss_datafunctiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
    where dt.maintable_id =  obj.object_id
    and objc.object_id = dtd.table_id
    and dt.object_id = dtd.object_id
)
and object_fullname not like '[Unknown%'
and object_language_name != '<N/A>'
and object_language_name != 'N/A'
and object_type_str not like  '%Project'
and object_type_str not like  '%Directory'
and object_type_str not like  '%Folder'
and object_type_str not in ('Cobol Paragraph','Cobol Section','Cobol CopyBook','Cobol Data Link','Cobol Entry
Point')
 order by 2,1;
```

The result should look like this:



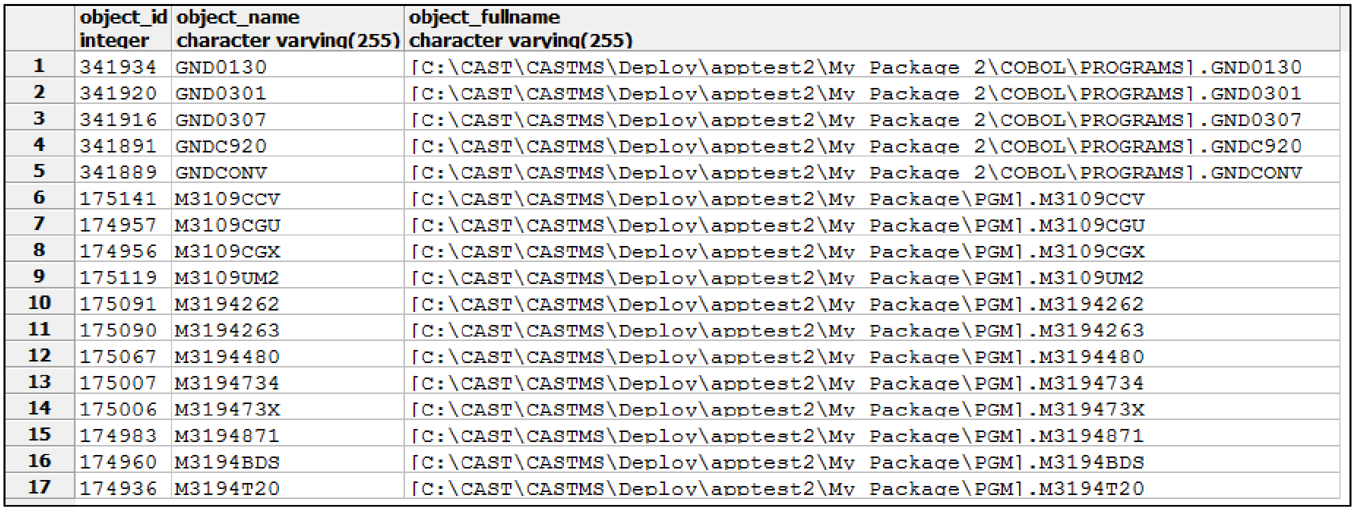
	object_id	object_name	object_fullname	object_type_str	object_language_name
	integer	character varying(255)	character varying(255)	character varying(255)	character varying(255)
46	196571	CYC-IN	[C:\CAST\CASTMS\Deploy\	Cobol File Link	Cobol
47	196572	DET-IN	[C:\CAST\CASTMS\Deploy\	Cobol File Link	Cobol
48	196581	EPAY-ERR	[C:\CAST\CASTMS\Deploy\	Cobol File Link	Cobol
49	196583	EPAY-IN	[C:\CAST\CASTMS\Deploy\	Cobol File Link	Cobol
50	196569	EPAY-OUT	[C:\CAST\CASTMS\Deploy\	Cobol File Link	Cobol
51	175169	FABSCTL7	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
52	341934	GND0130	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
53	341920	GND0301	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
54	341916	GND0307	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
55	341863	GND105ID	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
56	341860	GND110ID	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
57	341858	GND130IM	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
58	341855	GND150IQ	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
59	341853	GND200IM	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
60	341847	GND200IW	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
61	341834	GND210IW	[C:\CAST\CASTMS\Deploy\	JCL Step	JCL
62	341896	GNDBRCR	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
63	341895	GNDBRCRS	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
64	341894	GNDC900	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
65	341893	GNDC905	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol
66	341892	GNDC910	[C:\CAST\CASTMS\Deploy\	Cobol Program	Cobol

Objects that do not contribute to any transaction and that are not called by another object

The following SQL query will extract the objects that are not part of any transaction. This is illustrated with COBOL in the following example but the query can be adapted to other object types:

```
set search_path=<prefix>_local;

select obj.object_id,obj.object_name,obj.object_fullname
from CDT_OBJECTS obj
where obj.object_type_str = 'Cobol Program'
and object_fullname not like '[Unknown%'
and obj.object_id not in (      --- reduce the list to the program which are not part of a transaction
select distinct objc.object_id
from dss_transaction dt, dss_transactiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
where dt.form_id =  obj.object_id
and objc.object_id = dtd.child_id
and dt.object_id = dtd.object_id
union
select distinct objc.object_id
from dss_datafunction dt, dss_datafunctiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
where dt.maintable_id =  obj.object_id
and objc.object_id = dtd.table_id
and dt.object_id = dtd.object_id)
and obj.object_id not in (--- reduce the list to the program which are not called by something else
select obj.object_id
from ctv_links cl,CDT_OBJECTS obj
where cl.called_id = obj.object_id
and obj.object_type_str = 'Cobol Program'
and object_fullname not like '[Unknown%'
)
order by 3;
```

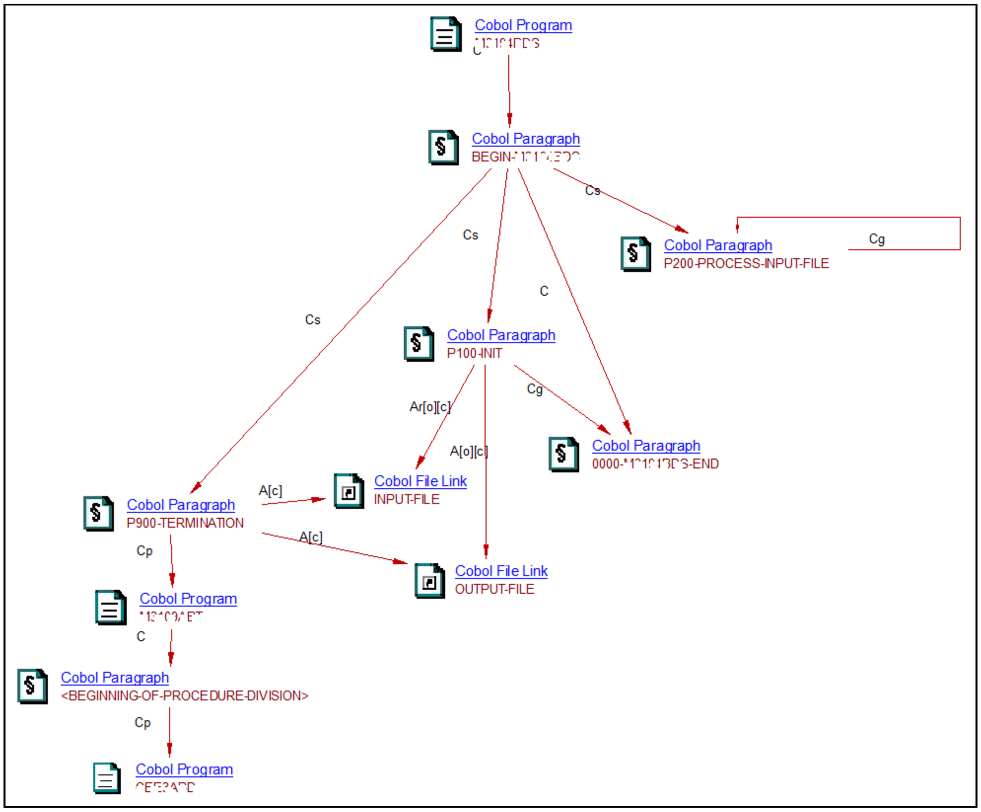
The following image shows the result set returned by the query:



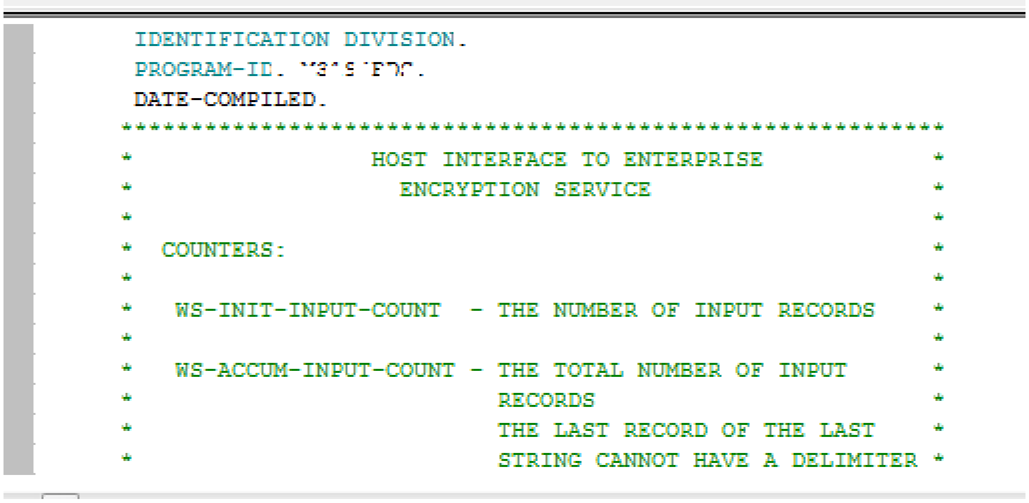
	object_id	object_name	object_fullname
	integer	character varying(255)	character varying(255)
1	341934	GND0130	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GND0130
2	341920	GND0301	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GND0301
3	341916	GND0307	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GND0307
4	341891	GNDC920	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GNDC920
5	341889	GNDCONV	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GNDCONV
6	175141	M3109CCV	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3109CCV
7	174957	M3109CGU	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3109CGU
8	174956	M3109CGX	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3109CGX
9	175119	M3109UM2	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3109UM2
10	175091	M3194262	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194262
11	175090	M3194263	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194263
12	175067	M3194480	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194480
13	175007	M3194734	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194734
14	175006	M319473X	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M319473X
15	174983	M3194871	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194871
16	174960	M3194BDS	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194BDS
17	174936	M3194T20	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194T20

It is is important to investigate this list. The objects with their associated links can be visualized in CAST Enlighten, as shown below:

Click to enlarge:



A good way to investigate is to look at the comments inserted in the source code as well as discussing with the SME:



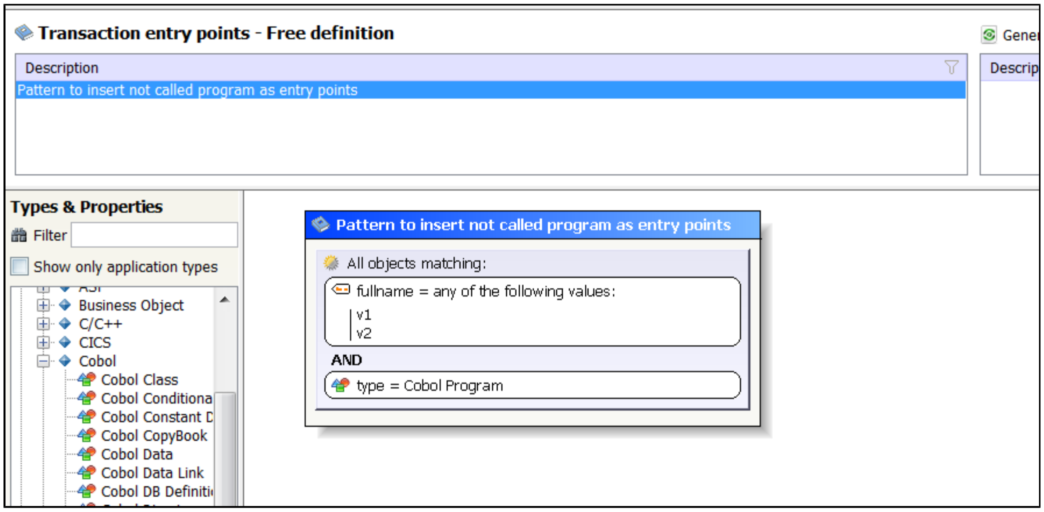
```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. [illegible].
       DATE-COMPILED.
      *****************************************************************
      *                 HOST INTERFACE TO ENTERPRISE                  *
      *                   ENCRYPTION SERVICE                          *
      *                                                               *
      *  COUNTERS:                                                    *
      *                                                               *
      *   WS-INIT-INPUT-COUNT  - THE NUMBER OF INPUT RECORDS          *
      *                                                               *
      *   WS-ACCUM-INPUT-COUNT - THE TOTAL NUMBER OF INPUT            *
      *                          RECORDS                              *
      *                          THE LAST RECORD OF THE LAST          *
      *                          STRING CANNOT HAVE A DELIMITER       *
```

In the above source code extract, we can see the program is an interface belonging to another application - it can therefore be considered as a **transaction end point**.

Adding new Transaction Entry Points

Use Free Definition rules in the CAST Transaction Configuration Center to select objects that should be considered as Transaction Entry Points (seeTransaction configuration for mote information). The following example uses COBOL programs but can be adapted to other types of objects as well:

Click to enlarge:



Description: Pattern to insert not called program as entry points

Pattern to insert not called program as entry points

All objects matching:

fullname = any of the following values:
v1
v2

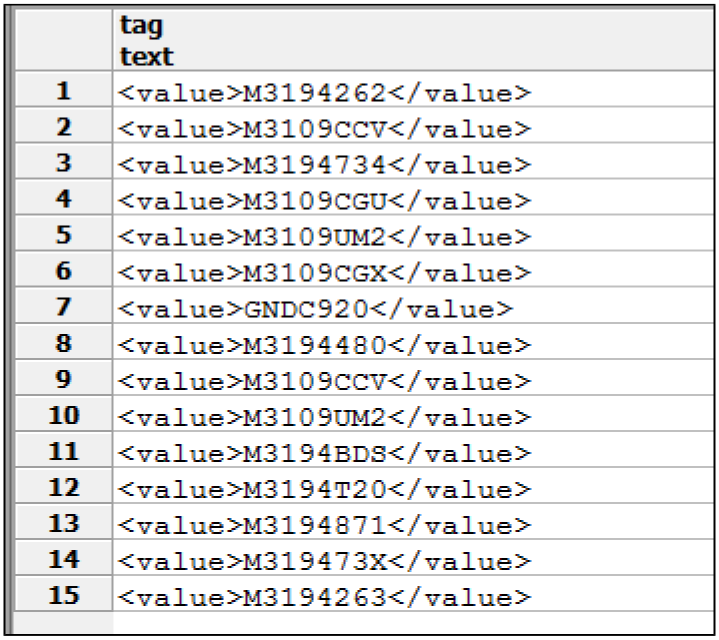
AND

type = Cobol Program

The objective here is to add new program names to the regular expression set up by the above rule in a limited number of operations. The first thing to do is to collect the program names you want to add to the regular expression. This list can be built by executing the following SQL query:

```
set search_path=<Prefix>_local;

select '<value>'||obj.object_name||'</value>' as text
from CDT_OBJECTS obj
where obj.object_type_str = 'Cobol Program'
and object_fullname not like '[Unknown%'
and obj.object_id not in ( --- reduce the list to the program which are not part of a transaction
select distinct objc.object_id
from dss_transaction dt, dss_transactiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
where dt.form_id = obj.object_id
and objc.object_id = dtd.child_id
and dt.object_id = dtd.object_id
union
select distinct objc.object_id
from dss_datafunction dt, dss_datafunctiondetails dtd, CDT_OBJECTS obj, cdt_objects objc
where dt.maintable_id = obj.object_id
and objc.object_id = dtd.table_id
and dt.object_id = dtd.object_id)
and obj.object_id not in (--- reduce the list to the program which are not called by something else
select obj.object_id
from ctv_links cl,CDT_OBJECTS obj
where cl.called_id = obj.object_id
and obj.object_type_str = 'Cobol Program'
and object_fullname not like '[Unknown%'
);
```

The result should look like this:



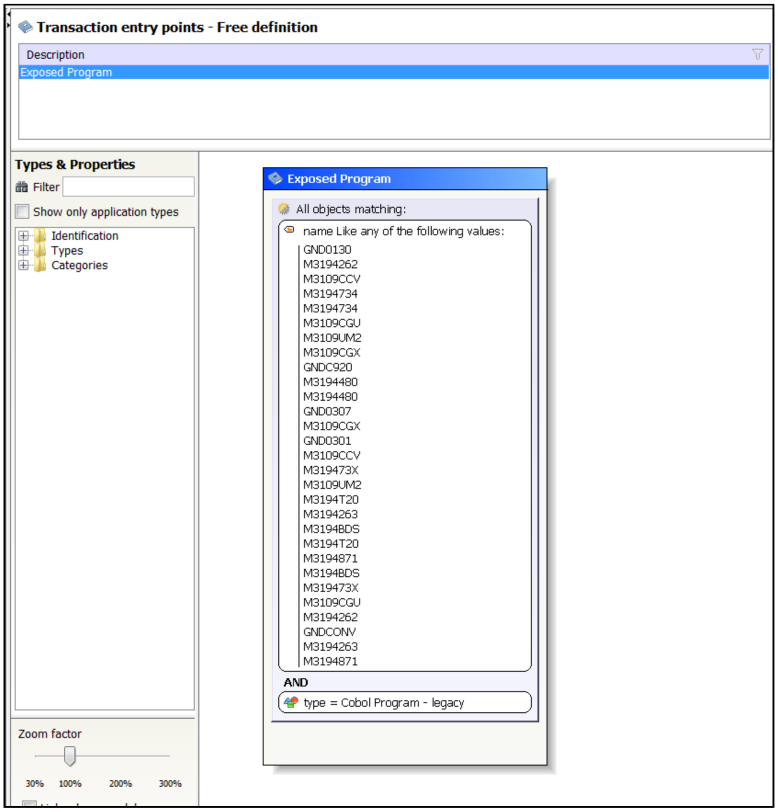
	tag text
1	<value>M3194262</value>
2	<value>M3109CCV</value>
3	<value>M3194734</value>
4	<value>M3109CGU</value>
5	<value>M3109UM2</value>
6	<value>M3109CGX</value>
7	<value>GNDC920</value>
8	<value>M3194480</value>
9	<value>M3109CCV</value>
10	<value>M3109UM2</value>
11	<value>M3194BDS</value>
12	<value>M3194T20</value>
13	<value>M3194871</value>
14	<value>M319473X</value>
15	<value>M3194263</value>

The second thing to do is to create an empty Transaction Entry Point rule and save it. You can call it "Exposed Program" for example.

The third operation is to replace the regular expression used in the Management Service rule by executing the following SQL query:

```
set search_path=<Prefix>_mngt;

update cal_objsetdef set setdefinition = '<set>
  <selection-criteria subobjects="no" externalobjects="yes">
   <property name = "name" operator = "eq" >
    <value>GND0130</value>
    <value>M3194262</value>
    <value>M3109CCV</value>
    <value>M3194734</value>
    <value>M3194734</value>
    <value>M3109CGU</value>
    <value>M3109UM2</value>
    <value>M3109CGX</value>
    <value>GNDC920</value>
    <value>M3194480</value>
    <value>M3194480</value>
    <value>GND0307</value>
    <value>M3109CGX</value>
    <value>GND0301</value>
    <value>M3109CCV</value>
    <value>M319473X</value>
    <value>M3109UM2</value>
    <value>M3194T20</value>
    <value>M3194263</value>
    <value>M3194BDS</value>
    <value>M3194T20</value>
    <value>M3194871</value>
    <value>M3194BDS</value>
    <value>M319473X</value>
    <value>M3109CGU</value>
    <value>M3194262</value>
    <value>GNDCONV</value>
    <value>M3194263</value>
    <value>M3194871</value>
   </property>
   <property name = "type" operator = "eq" >
    <value>CAST_COBOL_SavedProgram</value>
   </property>
  </selection-criteria>
</set>
' where setname = 'Exposed Program';
```

The result can be seen in TCC as shown below:



Transaction entry points - Free definition

Description
Exposed Program

Types & Properties

Filter

□ Show only application types

- Identification
- Types
- Categories

Exposed Program

All objects matching:

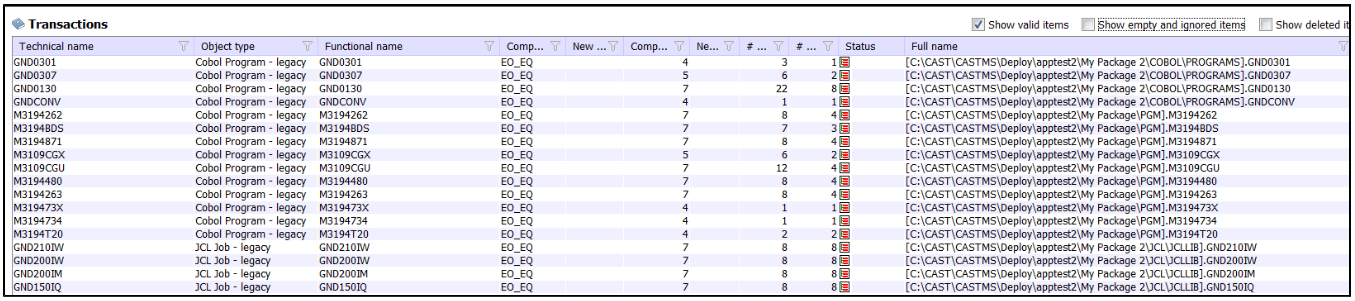
name Like any of the following values:

GND0130
M3194262
M3109CCV
M3194734
M3194734
M3109CGU
M3109UM2
M3109CGX
GNDC920
M3194480
M3194480
GND0307
M3109CGX
GND0301
M3109CCV
M319473X
M3109UM2
M3194T20
M3194263
M3194BDS
M3194T20
M3194871
M3194BDS
M319473X
M3109CGU
M3194262
GNDCONV
M3194263
M3194871

AND

type = Cobol Program - legacy

The Function Ppoint computation can then be done again and the expected transactions should appear:

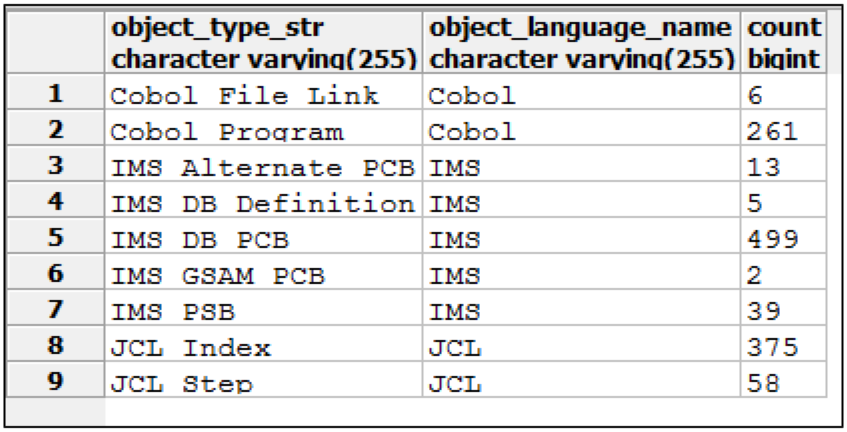


Transactions

☑ Show valid items □ Show empty and ignored items □ Show deleted it

Technical name	Object type	Functional name	Comp...	New ...	Comp...	Ne...	# ...	# ...	Status	Full name
GND0301	Cobol Program - legacy	GND0301	EO_EQ		4		3	1	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GND0301
GND0307	Cobol Program - legacy	GND0307	EO_EQ		5		6	2	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GND0307
GND0130	Cobol Program - legacy	GND0130	EO_EQ		7		22	8	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GND0130
GNDCONV	Cobol Program - legacy	GNDCONV	EO_EQ		4		1	1	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\COBOL\PROGRAMS].GNDCONV
M3194262	Cobol Program - legacy	M3194262	EO_EQ		7		8	4	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194262
M3194BDS	Cobol Program - legacy	M3194BDS	EO_EQ		7		7	3	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194BDS
M3194871	Cobol Program - legacy	M3194871	EO_EQ		7		8	4	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194871
M3109CGX	Cobol Program - legacy	M3109CGX	EO_EQ		5		6	2	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3109CGX
M3109CGU	Cobol Program - legacy	M3109CGU	EO_EQ		7		12	4	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3109CGU
M3194480	Cobol Program - legacy	M3194480	EO_EQ		7		8	4	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194480
M3194263	Cobol Program - legacy	M3194263	EO_EQ		7		8	4	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194263
M319473X	Cobol Program - legacy	M319473X	EO_EQ		4		1	1	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M319473X
M3194734	Cobol Program - legacy	M3194734	EO_EQ		4		1	1	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194734
M3194T20	Cobol Program - legacy	M3194T20	EO_EQ		4		2	2	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package\PGM].M3194T20
GND210IW	JCL Job - legacy	GND210IW	EO_EQ		7		8	8	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\JCL\JCLLIB].GND210IW
GND200IW	JCL Job - legacy	GND200IW	EO_EQ		7		8	8	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\JCL\JCLLIB].GND200IW
GND200IM	JCL Job - legacy	GND200IM	EO_EQ		7		8	8	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\JCL\JCLLIB].GND200IM
GND150IQ	JCL Job - legacy	GND150IQ	EO_EQ		7		8	8	▤	[C:\CAST\CASTMS\Deploy\apptest2\My Package 2\JCL\JCLLIB].GND150IQ

You can repeat to check if there are other objects that are not part of a transaction:

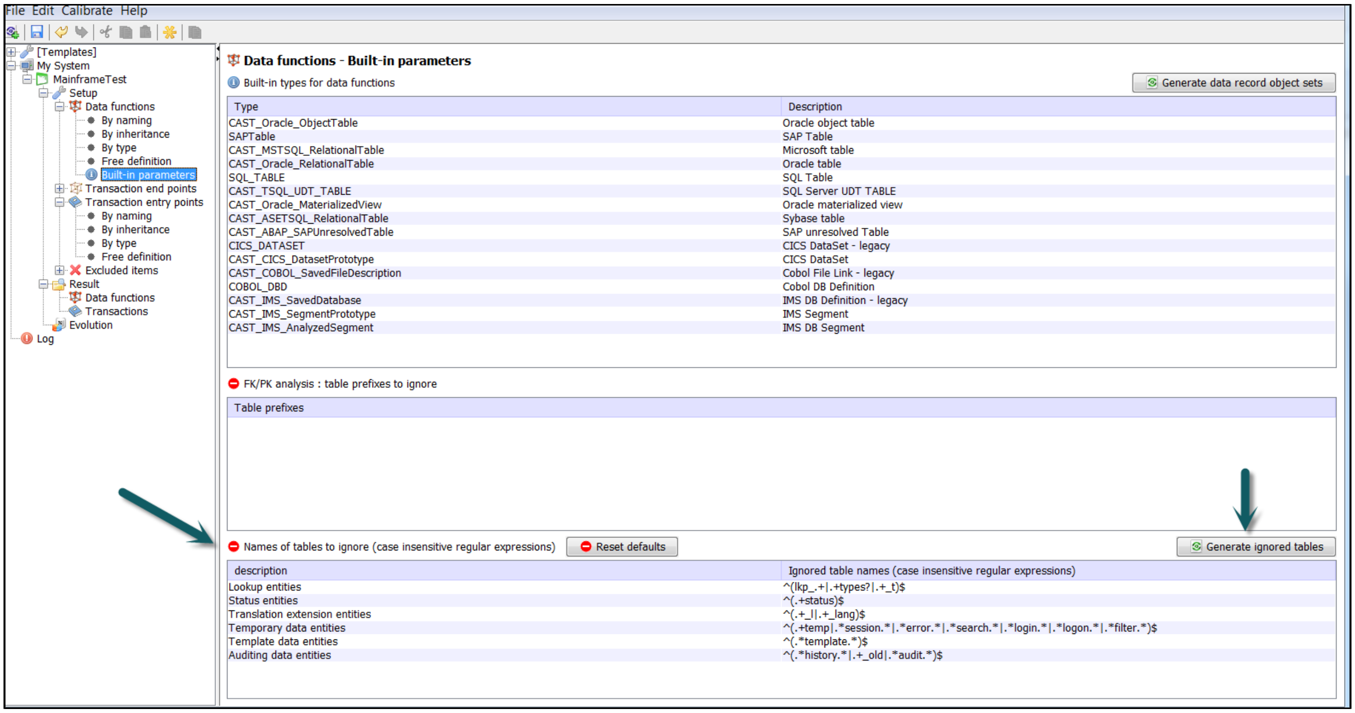


	object_type_str character varying(255)	object_language_name character varying(255)	count bigint
1	Cobol File Link	Cobol	6
2	Cobol Program	Cobol	261
3	IMS Alternate PCB	IMS	13
4	IMS DB Definition	IMS	5
5	IMS DB PCB	IMS	499
6	IMS GSAM PCB	IMS	2
7	IMS PSB	IMS	39
8	JCL Index	JCL	375
9	JCL Step	JCL	58

Check the database tables that have been excluded

Some database tables are automatically excluded from the Function Point computation process by applying a set of dedicated rules. Excluded database tables will not be visible in deleted, ignored, or retained Data Functions and it is important to validate the list to avoid unexpected exclusions. Currently, the only way to get this list is to generate the object set from the CAST Transaction Configuration Center "Built-in parameters" view:

Click to enlarge:



Data functions - Built-in parameters

Built-in types for data functions

Type	Description
CAST_Oracle_ObjectTable	Oracle object table
SAPTable	SAP Table
CAST_MSTSQL_RelationalTable	Microsoft table
CAST_Oracle_RelationalTable	Oracle table
SQL_TABLE	SQL Table
CAST_TSQL_UDT_TABLE	SQL Server UDT TABLE
CAST_Oracle_MaterializedView	Oracle materialized view
CAST_ASETSQL_RelationalTable	Sybase table
CAST_ABAP_SAPUnresolvedTable	SAP unresolved Table
CICS_DATASET	CICS DataSet - legacy
CAST_CICS_DatasetPrototype	CICS DataSet
CAST_COBOL_SavedFileDescription	Cobol File Link - legacy
COBOL_DBD	Cobol DB Definition
CAST_IMS_SavedDatabase	IMS DB Definition - legacy
CAST_IMS_SegmentPrototype	IMS Segment
CAST_IMS_AnalyzedSegment	IMS DB Segment

FK/PK analysis : table prefixes to ignore

Table prefixes

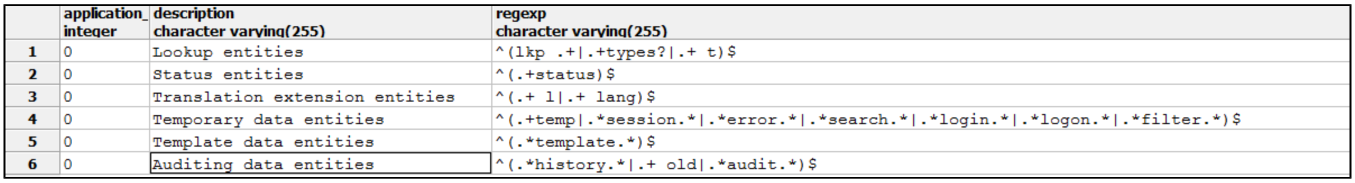
Names of tables to ignore (case insensitive regular expressions)

description	Ignored table names (case insensitive regular expressions)
Lookup entities	^(lkp_.+|.+types?|.+_t)$
Status entities	^(.+status)$
Translation extension entities	^(.+_l|.+_lang)$
Temporary data entities	^(.+temp|.*session.*|.*error.*|.*search.*|.*login.*|.*logon.*|.*filter.*)$
Template data entities	^(.*template.*)$
Auditing data entities	^(.*history.*|.+_old|.*audit.*)$

You can extract the list of exclusion rules with the following SQL query:

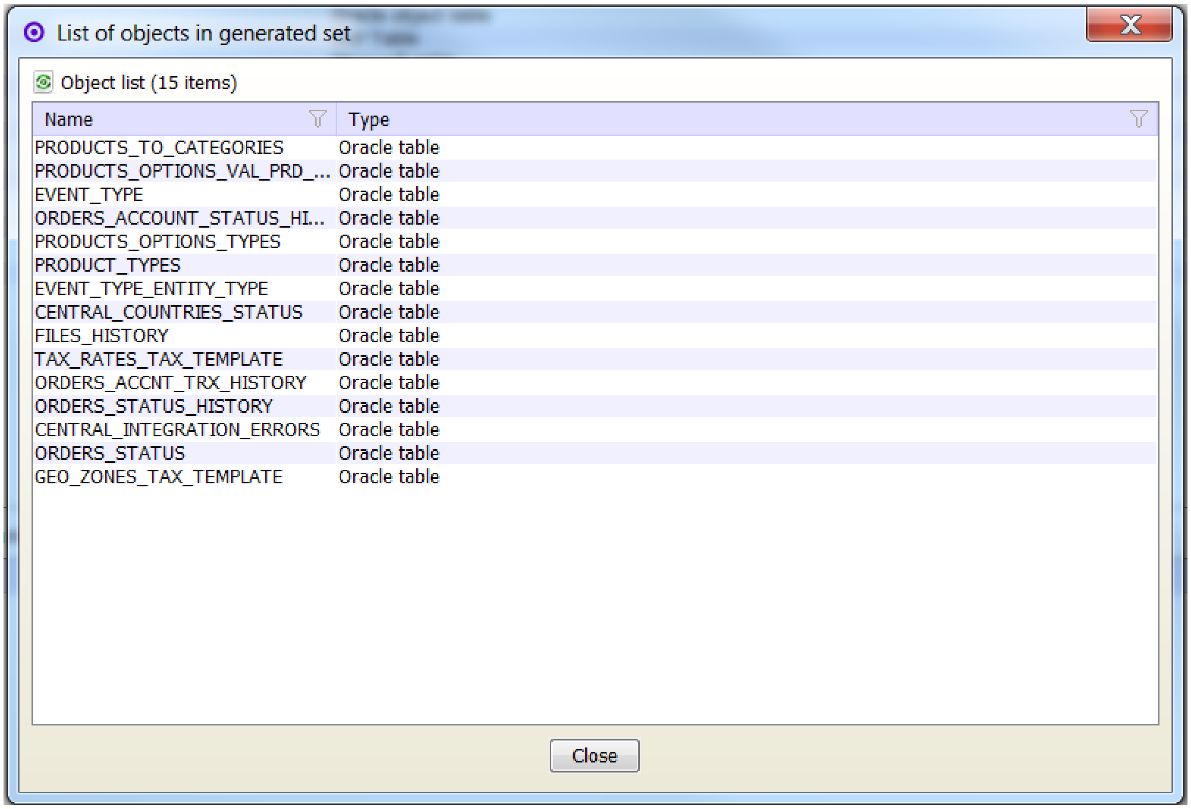
```
set search_path=<prefix>_mngt;
select * from cal_ignoredtable;
```

The result should look like this:



	application_ integer	description character varying(255)	regexp character varying(255)
1	0	Lookup entities	^(lkp .+|.+types?|.+ t)$
2	0	Status entities	^(.+status)$
3	0	Translation extension entities	^(.+ l|.+ lang)$
4	0	Temporary data entities	^(.+temp|.*session.*|.*error.*|.*search.*|.*login.*|.*logon.*|.*filter.*)$
5	0	Template data entities	^(.*template.*)$
6	0	Auditing data entities	^(.*history.*|.+ old|.*audit.*)$

The database tables selected by the exclusion rule are visible when you generate the object set in the CAST Transaction Configuration Center:



List of objects in generated set

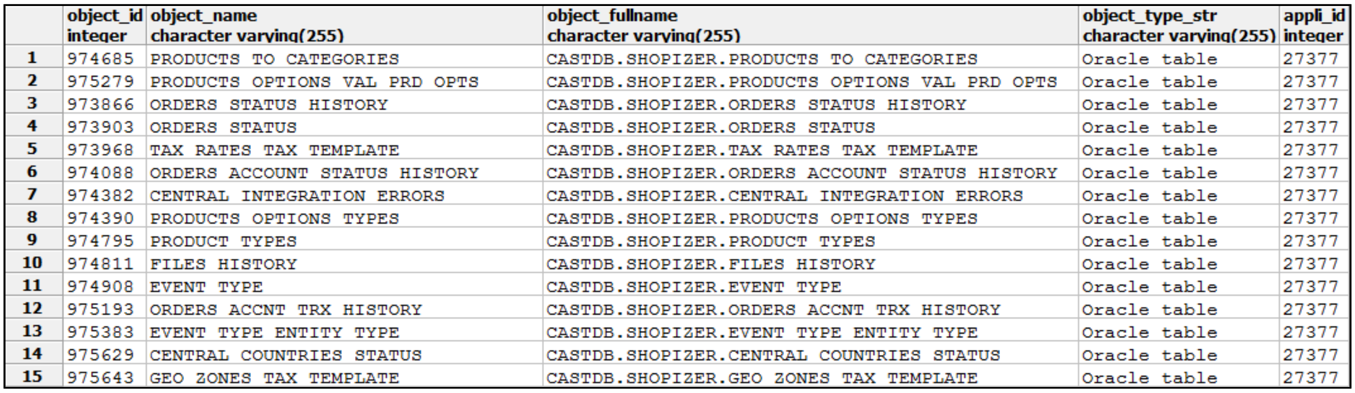
Object list (15 items)

Name	Type
PRODUCTS_TO_CATEGORIES	Oracle table
PRODUCTS_OPTIONS_VAL_PRD_...	Oracle table
EVENT_TYPE	Oracle table
ORDERS_ACCOUNT_STATUS_HI...	Oracle table
PRODUCTS_OPTIONS_TYPES	Oracle table
PRODUCT_TYPES	Oracle table
EVENT_TYPE_ENTITY_TYPE	Oracle table
CENTRAL_COUNTRIES_STATUS	Oracle table
FILES_HISTORY	Oracle table
TAX_RATES_TAX_TEMPLATE	Oracle table
ORDERS_ACCNT_TRX_HISTORY	Oracle table
ORDERS_STATUS_HISTORY	Oracle table
CENTRAL_INTEGRATION_ERRORS	Oracle table
ORDERS_STATUS	Oracle table
GEO_ZONES_TAX_TEMPLATE	Oracle table

These database tables can be also listed by applying the following SQL query:

```
set search_path=<prefix>_local;

Select obj.object_id, obj.object_name, obj.object_fullname, obj.object_type_str, fplt.appli_id
from fp_lookup_tables fplt, cdt_objects obj
where obj.object_id = fplt.object_id;
```

The result should look like this:



	object_id integer	object_name character varying(255)	object_fullname character varying(255)	object_type_str character varying(255)	appli_id integer
1	974685	PRODUCTS TO CATEGORIES	CASTDB.SHOPIZER.PRODUCTS TO CATEGORIES	Oracle table	27377
2	975279	PRODUCTS OPTIONS VAL PRD OPTS	CASTDB.SHOPIZER.PRODUCTS OPTIONS VAL PRD OPTS	Oracle table	27377
3	973866	ORDERS STATUS HISTORY	CASTDB.SHOPIZER.ORDERS STATUS HISTORY	Oracle table	27377
4	973903	ORDERS STATUS	CASTDB.SHOPIZER.ORDERS STATUS	Oracle table	27377
5	973968	TAX RATES TAX TEMPLATE	CASTDB.SHOPIZER.TAX RATES TAX TEMPLATE	Oracle table	27377
6	974088	ORDERS ACCOUNT STATUS HISTORY	CASTDB.SHOPIZER.ORDERS ACCOUNT STATUS HISTORY	Oracle table	27377
7	974382	CENTRAL INTEGRATION ERRORS	CASTDB.SHOPIZER.CENTRAL INTEGRATION ERRORS	Oracle table	27377
8	974390	PRODUCTS OPTIONS TYPES	CASTDB.SHOPIZER.PRODUCTS OPTIONS TYPES	Oracle table	27377
9	974795	PRODUCT TYPES	CASTDB.SHOPIZER.PRODUCT TYPES	Oracle table	27377
10	974811	FILES HISTORY	CASTDB.SHOPIZER.FILES HISTORY	Oracle table	27377
11	974908	EVENT TYPE	CASTDB.SHOPIZER.EVENT TYPE	Oracle table	27377
12	975193	ORDERS ACCNT TRX HISTORY	CASTDB.SHOPIZER.ORDERS ACCNT TRX HISTORY	Oracle table	27377
13	975383	EVENT TYPE ENTITY TYPE	CASTDB.SHOPIZER.EVENT TYPE ENTITY TYPE	Oracle table	27377
14	975629	CENTRAL COUNTRIES STATUS	CASTDB.SHOPIZER.CENTRAL COUNTRIES STATUS	Oracle table	27377
15	975643	GEO ZONES TAX TEMPLATE	CASTDB.SHOPIZER.GEO ZONES TAX TEMPLATE	Oracle table	27377

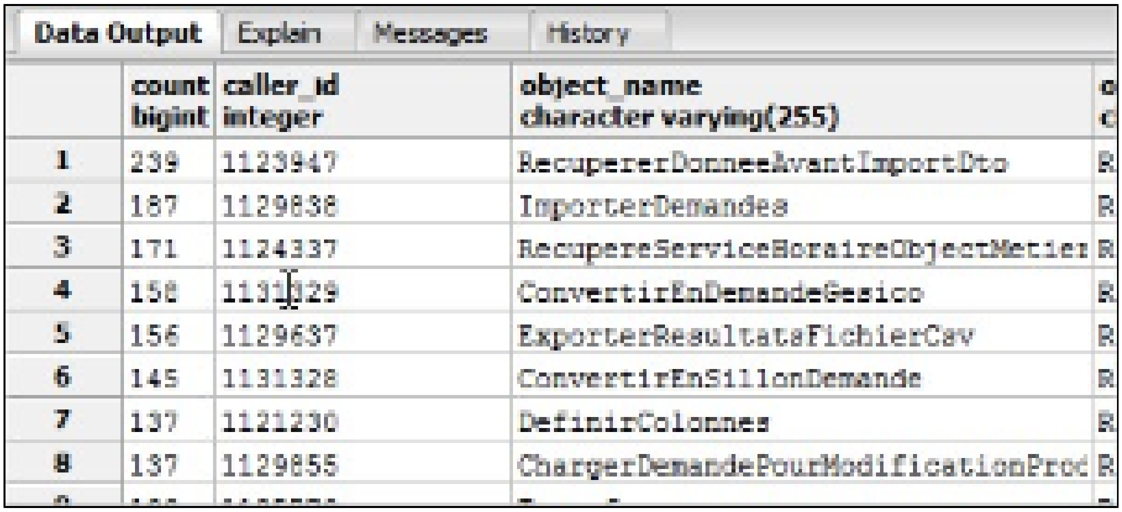
Appendix A: Checking links in call graphs

Objects with High Fan-Out

The purpose is to identify incorrect links OR side effects in the Function Point counting due to these links. The SQL query presented here identifies objects with High Fan-Out (objects that call a lots of other objects) and must be executed against the Analysis Service. You can customize it with snapshot_id, application_id, ... as well as the type of objects (in this example, "C# Method" are considered but you can search for "Java Method" in Java application as well).

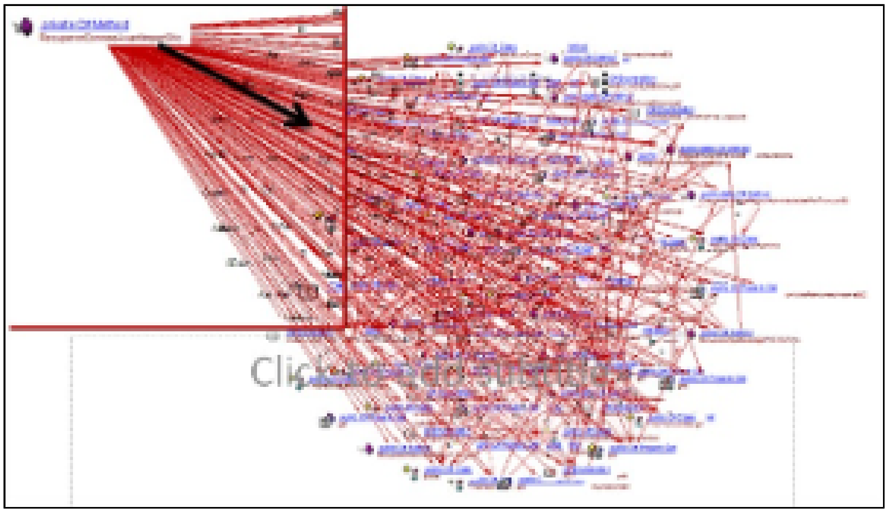
```
set search_path=<prefix>_local;

select count(L.caller_id), L.caller_id  , O.object_name, O.object_fullname
from ctv_links L, ctv_guid_objects O
where L.caller_id = O.object_id
and O.object_type_str = 'C# Method' --Java Method
group by L.caller_id,O.object_name,O.object_fullname
order by 1 desc
limit 100;
```

The results should look like this:



	count bigint	caller_id integer	object_name character varying(255)	o c
1	239	1123947	RecupererDonneeAvantImportDto	R
2	187	1129838	ImporterDemandes	R
3	171	1124337	RecupereServiceHoraireObjectMetier	R
4	158	1131329	ConvertirEnDemandeGesico	R
5	156	1129637	ExporterResultatsFichierCsv	R
6	145	1131328	ConvertirEnSillonDemande	R
7	137	1121230	DefinirColonnes	R
8	137	1129855	ChargerDemandePourModificationProd	R
9	[illegible]	[illegible]	[illegible]	[illegible]

In CAST Enlighten, these of objects can lead to this type of graphical view:

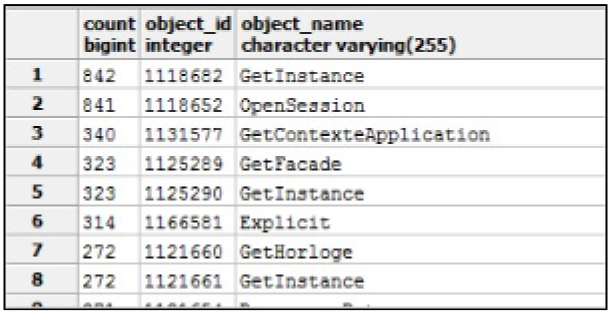


Objects with High Fan-In

The purpose is to identify wrong links OR side effects in the FP counting due to these links. The SQL query presented here identifies the objects with High Fan-In (objects that are called by a lot of other objects) and must be executed against the Analysis Service. You can customize it with snapshot_id, application_id, ... as well as the type of objects (in this example, "C# Method" as considered but you can search for "Java Method" in Java application as well).

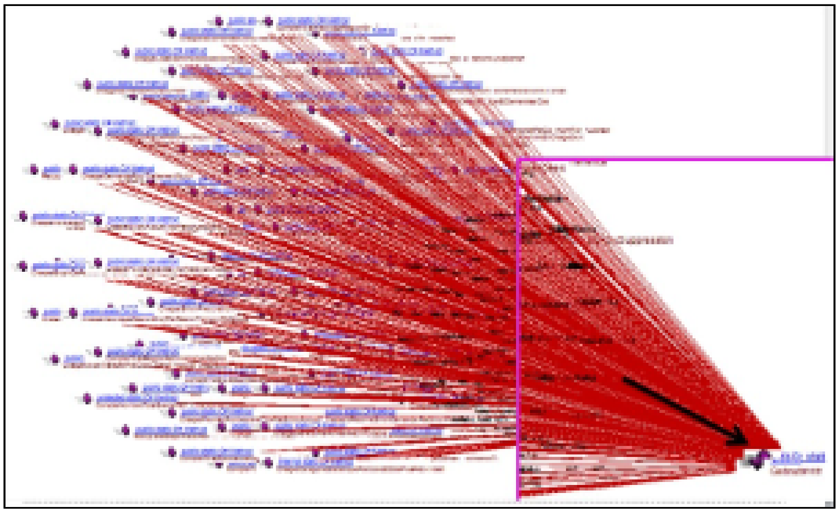
```
set search_path=<prefix>_local;

select count(O.object_id),O.object_id, O.object_name, O.object_fullname
from ctv_links L, ctv_guid_objects O
where L.called_id = O.object_id
and O.object_type_str = 'C# Method' --Java Method
group by L.called_id,O.object_id, O.object_name,O.object_fullname
order by 1 desc
limit 100;
```

The results should look like this:



	count bigint	object_id integer	object_name character varying(255)
1	842	1118682	GetInstance
2	841	1118652	OpenSession
3	340	1131577	GetContexteApplication
4	323	1125289	GetFacade
5	323	1125290	GetInstance
6	314	1166581	Explicit
7	272	1121660	GetHorloge
8	272	1121661	GetInstance

In CAST Enlighten, these of objects can lead to this type of graphical view:



Appendix B: Searching for potential Transaction Entry Points

The purpose is to identify missing links. The SQL query presented here identifies objects that are not called (ex: unreferenced methods) and must be executed against the Analysis Service. It can be customized with other types of objects (ex: "Java Method"):

```
set search_path=<prefix>_local;

select *
from ctv_guid_objects O
where O.object_id not in (select l.called_id from ctv_links L)
and O.object_type_str = 'C# Method'
limit 100;
```

This list must be validated by the SME and the application team and the transaction configuration must then be adjusted accordingly.

Appendix C: Searching for potential Transaction End Points

The purpose is to identify missing links. The SQL query presented here identifies objects that do not call any other object. This situation may happen if, for instance, a DAO object is analyzed and there is no corresponding database in the application boundary. This object can be seen as an interface and then it should be considered as Transaction End Point that contributes to the transaction.

This SQL query must be executed against the Analysis Service and can be customized with other types of objects (ex: "Java Method"):

```
set search_path=<prefix>_local;

select *
from ctv_guid_objects O
where O.object_id not in (select l.caller_id from ctv_links L)
and O.object_type_str = 'C# Method'
limit 100;
```

This list must be validated by the SME and the application team and the transaction configuration must be adjusted accordingly.

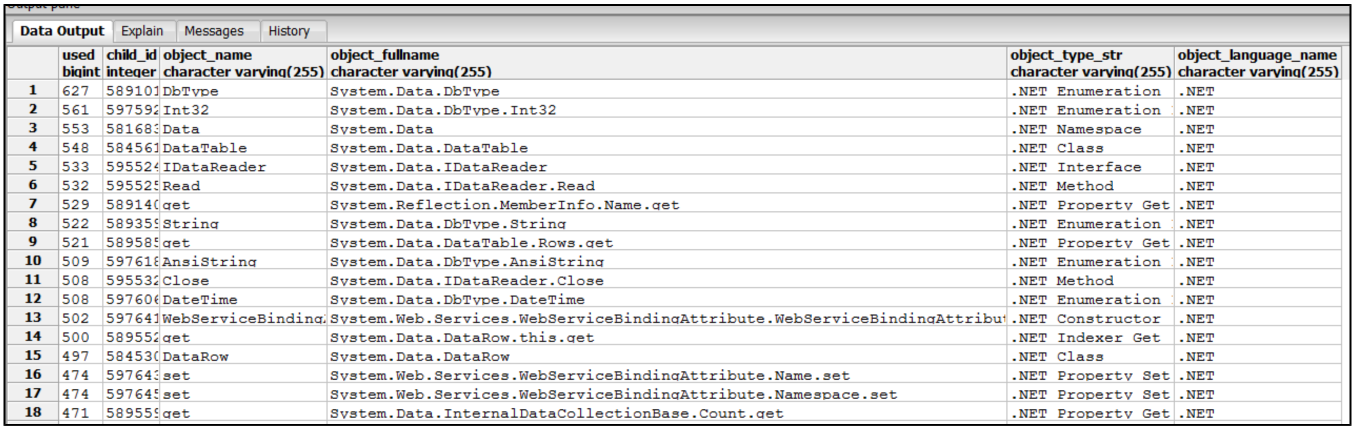
Appendix D: List of Transaction End Points

The following SQL query searches for all the objects that have been identified as Transaction End Points. It must be executed against the Analsysis Service.

```
set search_path=<prefix>_local;

select count(1) as used, dtd.child_id, obj.object_name, obj.object_fullname, obj.object_type_str, obj.
object_language_name
from dss_transactiondetails dtd, cdt_objects obj
where dtd.childtype in (5, 6 ,7)
and dtd.child_id = obj.object_id
group by 2,3,4,5,6
order by 1 desc,2 asc ,3 asc;
```

The results should look like this:

Click to enlarge:



	used bigint	child_id integer	object_name character varying(255)	object_fullname character varying(255)	object_type_str character varying(255)	object_language_name character varying(255)
1	627	589101	DbType	System.Data.DbType	.NET Enumeration	.NET
2	561	597592	Int32	System.Data.DbType.Int32	.NET Enumeration	.NET
3	553	581683	Data	System.Data	.NET Namespace	.NET
4	548	584561	DataTable	System.Data.DataTable	.NET Class	.NET
5	533	595524	IDataReader	System.Data.IDataReader	.NET Interface	.NET
6	532	595525	Read	System.Data.IDataReader.Read	.NET Method	.NET
7	529	589140	get	System.Reflection.MemberInfo.Name.get	.NET Property Get	.NET
8	522	589359	String	System.Data.DbType.String	.NET Enumeration	.NET
9	521	589585	get	System.Data.DataTable.Rows.get	.NET Property Get	.NET
10	509	597618	AnsiString	System.Data.DbType.AnsiString	.NET Enumeration	.NET
11	508	595532	Close	System.Data.IDataReader.Close	.NET Method	.NET
12	508	597606	DateTime	System.Data.DbType.DateTime	.NET Enumeration	.NET
13	502	597641	WebServiceBinding	System.Web.Services.WebServiceBindingAttribute.WebServiceBindingAttribut	.NET Constructor	.NET
14	500	589552	get	System.Data.DataRow.this.get	.NET Indexer Get	.NET
15	497	584530	DataRow	System.Data.DataRow	.NET Class	.NET
16	474	597643	set	System.Web.Services.WebServiceBindingAttribute.Name.set	.NET Property Set	.NET
17	474	597645	set	System.Web.Services.WebServiceBindingAttribute.Namespace.set	.NET Property Set	.NET
18	471	589559	get	System.Data.InternalDataCollectionBase.Count.get	.NET Property Get	.NET

This list must be validated by the SME and the application team.