Transactional function completeness

On this page:

- Introduction
- Collect the objects that have been removed from the computation
  - Excluded Data Entities
  - Deleted or ignored Transactional Functions
  - Deleted or ignored Data Functions
- Computed Function Points
  - Function Points for Data Functions
  - Function Points for Transactional Functions
  - Collect the Functions
  - Data Functions
  - Transactional Functions
- Information related to Transaction Configuration
  - List of Transaction candidates
- Information related to Transactional Function Calibration
  - Collect Transaction Entry Points that have been merged
  - Collect Transactional Functions that have been adjusted
  - Collect ignored or deleted Transactional Functions
  - Collect empty Transactional Functions not ignored or deleted
- Information related to Data Entities configuration
  - List of Data Entity candidates
- Information related to Data Functions calibration
  - Collect Data Functions that have been merged
  - Collect Data Functions that have been adjusted
  - Collect ignored or deleted Data Functions

Introduction

Once the Function Points calibration is completed and the counting has been done, you should communicate the results to the team with all the materials justifying how you got these results. The SQL queries presented in this section allow you to collect this information.

Collect the objects that have been removed from the computation

Excluded Data Entities

The following SQL query collects the database tables that have been excluded from the computation:

```
set search_path=<Prefix>_local;

select * from FP_Lookup_Tables a join CDT_OBJECTS b on a.object_id = b.object_id;
```

Deleted or ignored Transactional Functions

The following SQL query collects the Transactional Functions that have been deleted or ignored in the computation:

```
set search_path=<Prefix>_local;

select count (*) from dss_transaction t where t.appli_id in (select appli_id from fp_cms_application) and not
exists(select 1 from ctt_object_applications oa, fp_cms_application a where t.form_id = oa.object_id and oa.
application_id = a.module_id);
```

Deleted or ignored Data Functions

The following SQL query collects the Data Functions that have been deleted or ignored in the computation:

```
set search_path=<Prefix>_local;

select count (*) from dss_datafunction df where not exists(select 1 from ctt_object_applications oa,
fp_cms_application a where df.maintable_id = oa.object_id and oa.application_id = a.module_id);
```

Computed Function Points

The total number of Function Points for the application is the sum of the Function Points for Data Functions and Function Points for Transactional Functions. Each part of the total can be presented separately.

Function Points for Data Functions

The following SQL query provides the number of Data Functions that have been taken in to account in the counting process:

```
set search_path=<Prefix>_local;

select count(*) from dss_datafunction where cal_flags in (  0, 2 , 256 );
```

This number can be correlated to the number of Function Points that have been calculated. This second number is provided by the below SQL query:

```
set search_path=<Prefix>_local;

select sum(ilf_ex) from dss_datafunction where cal_flags in (  0, 2 , 256 );
```

Function Points for Transactional Functions

The following SQL query provides the number of Transactional Functions that have been taken in account in the counting:

```
set search_path=<Prefix>_local;

select count(*) from dss_transaction where cal_flags in (  0, 2 );
```

This number can be correlated to the number of Function Points that have been calculated. This second number is provided by the below SQL query:

```
set search_path=<Prefix>_local;

select sum(tf_ex) from dss_transaction where cal_flags in (  0, 2 );
```

Collect the Functions

Data Functions and Transactional Functions are the elements taken in to account in the counting process. They are part of the functional sizing deliverables.

Data Functions

The following SQL query provides the number of Data Functions that have been identified in the application and that contribute to its functional size:

```
set search_path=<Prefix>_local;

select count (1) as NumberOfSelectedLogicalFile from (
select cob.object_name as LogicalFile,
cob.object_fullname as LogicalFileFullname,
dtf.DET as DET,
dtf.RET as RET,
case dtf.isinternal when 0 then 'EIF' when 1 then 'ILF' END as DefaultType,
case dtf.user_isinternal when 0 then 'EIF' when 1 then 'ILF' END as OverwriteType,
dtf.ilf_ex as DefaultFPValue,
dtf.user_fp_value as OverwriteFPValue,
case dtf.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_datafunction dtf, cdt_objects cob
where dtf.maintable_id = cob.object_id
and dtf.cal_flags in (0,2)
order by 9 ASC) as result;
```

The result should look like this:

	numberofselectedlogicalfile bigint
1	1393

The following SQL query provides the associated list of Data Functions:

```
set search_path=<Prefix>_local;

select cob.object_name as LogicalFile,
cob.object_fullname as LogicalFileFullname,
dtf.DET as DET,
dtf.RET as RET,
case dtf.isinternal when 0 then 'EIF' when 1 then 'ILF' END as DefaultType,
case dtf.user_isinternal when 0 then 'EIF' when 1 then 'ILF' END as OverwriteType,
dtf.ilf_ex as DefaultFPValue,
dtf.user_fp_value as OverwriteFPValue,
case dtf.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_datafunction dtf, cdt_objects cob
where dtf.maintable_id = cob.object_id
and dtf.cal_flags in (0,2)
order by 9 ASC;
```

The result should look like this:

	logicalfile character varying(255)	logicalfilefullname character varying(255)	det integer	ret integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	status text
176	ContractDefinitions	default.DocArchDB.d	129	30	ILF		15		Root
177	axMoverSettings	default.DocArchDB.d	3	2	EIF		5		Root
178	axMoverMain	default.DocArchDB.d	3	2	EIF		5		Root
179	MSsavedforeignkeys	default.CorporateDa	12	7	EIF		7		Root
180	MSsavedforeignkeyex	default.CorporateDa	6	7	EIF		7		Root
181	AuditType	default.Adworks.dbo	19	4	ILF		7		Root
182	AuditTable	default.Adworks.dbo	14	3	ILF		7		Root
183	MSsavedforeignkeyco	default.CorporateDa	7	7	EIF		7		Root
184	Shipping Monitor	default.FreightDB.d	8	2	EIF		5		Root
185	MSreplication subsc	default.CorporateDa	14	7	EIF		7		Root
186	Attribute ProductAt	default.WebDB PDM.d	13	2	EIF		5		STD
187	Attribute ProductAt	default.WebDB PDM.d	31	4	EIF		7		STD

Transactional Functions

The following SQL query provides the number of Transactional Functions that are have been identified in the application and that contribute to its functional size:

```
set search_path=<Prefix>_local;

select count (1) as NumberOfSelectedEntry from (
select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue, DTR.cal_flags,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
order by 9 ASC, 2 ASC) as result;
```

The result should look like this:

	numberofselectedentry bigint
1	2073

The following SQL query provides the associated list of Transactional Functions:

```
set search_path=<Prefix>_local;

select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
order by 9 ASC, 2 ASC;
```

The result should look like this:

	transaction character varying(255)	transactionfullname character varying(255)	det integer	ftr integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	status text
1	Main	AS400DBCrawler.Prog	840	167	EI	EI	7		Root
2	Program	AS400FileCompare.Pr	5	1	EI	EI	4		Root
3	Utilities	BusinessObject.Util	200	40	EI	EI	7		Root
4	Paintbox	CallManager.ActiveG	0	0	EI	EI	0		Root
5	ActiveRow	CallManager.ActiveR	20	4	EI	EI	7		Root
6	ActiveCell	CallManager.ActiveR	0	0	EI	EI	0		Root
7	Forml	CallManager.Forml	30	6	EI	EI	7		Root
8	WaitForml	CallManager.WaitFor	0	0	EI	EI	0		Root

Information related to Transaction Configuration

It is helpful for the person who will receive the functional sizing results to know which objects have been considered during the configuration phase. The SQL queries presented here allow you to collect this information.

List of Transaction candidates

The SQL query presented below collects all the objects that can be considered as starting a transaction. Some of them have been excluded or ignored, others have been kept and led to Transactional Functions.

```
set search_path=<Prefix>_local;

select cob.object_name as Transaction,
cob.object_fullname as LogicalFileFullname,
cob.object_mangling,
cob.object_type_str
from dss_transaction dtf, cdt_objects cob
where dtf.form_id = cob.object_id
order by 3 ASC, 2 ASC;
```

The result should look like this:

Click to enlarge

	transaction character varying(255)	logicalfilefullname character varying(255)	object_mangling character varying(255)	object_type_str character varying(255)
1	Main	.BatchJobs.ServiceRequestQueueProcessor.Program.Main	Main()System.Int32	C# Method
2	Main	.Windows.BatchJobs.As400GenericDownloader.As400DownloadBatch.Main	Main()System.Int32	C# Method
3	Main	.Windows.BatchJobs.ContractArchiveManager.BatchProcessor.Main	Main()System.Int32	C# Method
4	Main	core.Sample.AppFacadeUI.Program.Main	Main()System.Void	C# Method
5	Main	AppCore.Sample.Audit.Gui.Program.Main	Main()System.Void	C# Method
6	Main	AppCore.Sample.DataContextUI.Program.Main	Main()System.Void	C# Method
7	Main	AppCore.Sample.UIControls.Program.Main	Main()System.Void	C# Method
8	Main	AppCore.Sample.Validation.Program.Main	Main()System.Void	C# Method
9	Main	OutboundCallClientWindowsApp.Program.Main	Main()System.Void	C# Method
10	Main	OutboundCallServerWindowsApp.Program.Main	Main()System.Void	C# Method
11	Main	OutboundCallManager.Program.Main	Main()System.Void	C# Method
12	Main	CallManager.Program.Main	Main()System.Void	C# Method
13	Main	CallMonitor.Program.Main	Main()System.Void	C# Method
14	Main	.CTI.InboundCallServiceMonitor.Program.Main	Main()System.Void	C# Method
15	Main	.InboundCallManagerService.Monitor.Program.Main	Main()System.Void	C# Method
16	Main	.IntApps.WinApp.CoffeeShop.Program.Main	Main()System.Void	C# Method
17	Main	.IntApps.WinApp.Supermarket.Program.Main	Main()System.Void	C# Method
18	Main	.ShippingLogMonitor.ShippingLogMonitorProgramMain.Main	Main()System.Void	C# Method
19	Main	.Windows.Applications.CdwAdmin.MainUI.Program.Main	Main()System.Void	C# Method

Information related to Transactional Function Calibration

It is helpful for the person who will receive the functional sizing results to know if adjustments have been done during the calibration phase. The SQL queries presented here allow you to collect this information.

Collect Transaction Entry Points that have been merged

The following SQL query collects Transactional Functions that have been merged during the calibration phase:

```
set search_path=<Prefix>_local;

select * from (
select dtr.object_id as form_id,
cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
union all
select dtr.cal_mergeroot_id as form_id,' |-------' || cob.object_name as transaction, cob.object_fullname as
transactionfullname, dtr.DET as DET, DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id > 0 -- not a sub transaction
and dtr.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
) as result
order by 1 ASC, 10 DESC;
```

The result should look like this:

	form_id integer	transaction character varying	transactionfullname character varying(255)	det integer	ftr integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	status text
1	1152580	AS400HealthWebService.asmx	S:\Source\ Source` lient\Source` BusinessObjects.Applicatio	5	1	EI	EI	4		STD
2	1152580	AS400InterfaceWS.asmx	S:\Source\ \Source\ lient\Source. nterprise.WebService\AS40	0	0	EI	EI	0		Root
3	1152580	|-------AS400InterfaceWS.asmx	S:\Source\ \Source\WebServices(Enterprise)\AS400DiagInfoWS\AS400In	0	0	EI	EI	0		Child
4	1152580	AccountCodingWS.asmx	S:\Source\ \Source` \Source Enterprise.WebService\Acco	10	2	EI	EI	5		Root
5	1152580	|-------AccountCodingWS.asmx	S:\Source\ \Source\WebServices(Enterprise)\AccountCodingWS\Account	10	2	EI	EI	5		Child
6	1152580	AddressValidationWS.asmx	S:\Source\ \Source nterprise.WebService\Addr	10	2	EI	EI	5		Root
7	1152580	|-------AddressValidationWS.asmx	S:\Source\ S\Source\WebServices(Enterprise)\AddressValidationWS\Add	10	2	EI	EI	5		Child
8	1152580	|-------AddressValidationRecord	.Windows.Applications. .BusinessObjects.AddressValidationRec	0	0	EI	EI	0		Child
9	1152580	|-------AddressValidationUtility	.Windows.Applications. .MainUI.AddressValidationUtility	0	0	EI	EI	0		Child

Collect Transactional Functions that have been adjusted

The following SQL query collects transactions for which the type or the Function Point value has been adjusted during the calibration phase:

```
set search_path=<Prefix>_local;

select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.user_isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and (DTR.user_fp_value >= 0 or DTR.user_isinput >= 0)
order by 2 ASC;
```

The result should look like this:

	transaction character varying(255)	transactionfullname character varying(255)	det integer	ftr integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer
31	ArchiveDetailsForm	Cdw.Windows.Applica	165	33	EI	EI	7	
32	ArchiveSearchContro	Cdw.Windows.Applica	155	31	EI	EI	7	
33	BidDetailController	Cdw.Windows.Applica	150	30	EI	EI	7	
34	BidDetailForm	Cdw.Windows.Applica	170	34	EI	EI	7	
35	BidLineSearchContro	Cdw.Windows.Applica	75	15	EI	EI	7	
36	BidLineSearchForm	Cdw.Windows.Applica	165	33	EI	EI	7	
37	BidSearchUIControll	Cdw.Windows.Applica	155	31	EI	EI	7	
38	BidSearchView	Cdw.Windows.Applica	85	17	EI	EI	7	
39	NoteListControl	Cdw.Windows.Applica	0	0	EI	EI	0	
40	CustomerSearchDialo	Cdw.Windows.Applica	0	0	EI	EI	0	
41	CRMCoWorkerSearch	Cdw.Windows.Applica	5	1	EI	EI	4	

Collect ignored or deleted Transactional Functions

The following SQL query collects the number of Transactional Functions that have been ignored or deleted during the calibration phase:

```
set search_path=<Prefix>_local;

select count (1) as NumberOfIgnoredOrDeletedEntry from (
select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue, DTR.cal_flags,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
order by 9 ASC, 2 ASC) as result;
```

The result should look like this:

	numberofignoredordeletedentry bigint
1	11

The next SQL query provides the associated list of Transactional Functions:

```
set search_path=<Prefix>_local;

select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
order by 9 ASC, 2 ASC;
```

The result should look like this:

	transaction character varying(255)	transactionfullname character varying(255)	det integer	ftr integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	status text
1	Main	AXCOMTester.Program	0	0	EI	EI	0		Deleted
2	Program	AdaptiveCacheTest.P	0	0	EI	EI	0		Deleted
3	Main	AdaptiveCacheTest.P	80	16	EI	EI	7		Deleted
4	Main	AppplicationXtender	20	4	EI	EI	7		Deleted
5	Program	AxWebServicesTester	0	0	EI	EI	0		Deleted
6	Main	AxWebServicesTester	15	3	EI	EI	7		Deleted
7	Main	CDW Tapi ServiceLib	5	1	EI	EI	4		Deleted
8	CodedUITest1	CTICodedUITests.Cod	0	0	EI	EI	0		Deleted
9	Main	CallHandlingTester.	0	0	EI	EI	0		Deleted
10	Form1	CallManager.Form1	30	6	EI	EI	7		Deleted
11	WaitForm1	CallManager.WaitFor	0	0	EI	EI	0		Deleted

Collect empty Transactional Functions not ignored or deleted

The following SQL query collects the Transactional Functions that are not complete but that have not been ignored or deleted during the calibration phase:

```
set search_path=<Prefix>_local;

select count (1) as NumberOfEmptyTransactionNotIgnoredOrDeletedEntry from (
select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue, DTR.cal_flags,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
and DTR.tf_ex=0
order by 9 ASC, 2 ASC) as result;
```

The result should look like this:

	numberofemptytransactionnotignoredordeletedentry bigint
1	738

The next SQL query provides the associated list of Transactional Functions:

```
set search_path=<Prefix>_local;

select cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtr.DET as DET,
DTR.FTR as FTR,
case DTR.isinput when 0 then 'EI' when 1 then 'EO_EQ' END as DefaultType,
case DTR.isinput when 0 then 'EI' when 1 then 'EO' when 2 then 'EQ' END as OverwriteType,
DTR.tf_ex as DefaultFPValue,
DTR.user_fp_value as OverwriteFPValue, DTR.cal_flags,
case DTR.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_transaction dtr, cdt_objects cob
where dtr.form_id = cob.object_id
and dtr.cal_mergeroot_id = 0 -- not a sub transaction
and dtr.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
and DTR.tf_ex=0
order by 9 ASC, 2 ASC;
```

The result should look like this:

	transaction character varying(255)	tr cl	det integer	ftr integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	cal_flags integer	status text
1	Program	Ar	0	0	EI	EI	0		0	STD
2	Program	Ar	0	0	EI	EI	0		0	STD
3	Main	Ar	0	0	EI	EI	0		0	STD
4	Program	Ar	0	0	EI	EI	0		0	STD
5	Program	Ar	0	0	EI	EI	0		0	STD
6	Main	Ar	0	0	EI	EI	0		0	STD
7	Program	Ar	0	0	EI	EI	0		0	STD
8	BasicWinFormProgr	Ba	0	0	EI	EI	0		0	STD
9	Main	Ba	0	0	EI	EI	0		0	STD
10	Main	Cl	0	0	EI	EI	0		0	STD
11	Program	Cl	0	0	EI	EI	0		0	STD
12	Program	Cl	0	0	EI	EI	0		0	STD
13	Main	Cl	0	0	EI	EI	0		0	STD
14	ActiveColumnHeade	Cl	0	0	EI	EI	0		0	STD
15	Paintbox	Cl	0	0	EI	EI	0		0	STD
16	ActiveGridItemCom	Cl	0	0	EI	EI	0		0	STD
17	ActiveCell	Cl	0	0	EI	EI	0		0	STD
18	ActiveCellFader	Cl	0	0	EI	EI	0		0	STD

Information related to Data Entities configuration

It is helpful for the person who will receive the functional sizing results to know which objects have been considered during the configuration phase. The SQL queries presented here allow you to collect this information.

List of Data Entity candidates

The following SQL query collects all the Data Entities that have been identified as potential Data Functions. Some of them have been ignored or deleted and others have been kept and led to Data Functions.

```
set search_path=<Prefix>_local;

select cob.object_name as logicalFile,
cob.object_fullname as LogicalFileFullname,
cob.object_mangling,
cob.object_type_str
from dss_datafunction dtf, cdt_objects cob
where dtf.maintable_id = cob.object_id
order by 3 ASC, 2 ASC;
```

The result should look like this:

	logicalfile character varying(255)	logicalfilefullname character varying(255)	object_mangling character varying(255)	object_type_str character varying(255)
1	ADD ITEM ACCE	default.AdvertisementsDB.dbo.ADD ITEM ACCE	N/A	Microsoft table
2	AD MEDI	default.AdvertisementsDB.dbo.AD MEDI	N/A	Microsoft table
3	AD MEDI PRIC	default.AdvertisementsDB.dbo.AD MEDI PRIC	N/A	Microsoft table
4	AD PRIC IMPORT	default.AdvertisementsDB.dbo.AD PRIC IMPORT	N/A	Microsoft table
5	CouponLog	default.AdvertisementsDB.dbo.CouponLog	N/A	Microsoft table
6	CouponMaster	default.AdvertisementsDB.dbo.CouponMaster	N/A	Microsoft table
7	DG ACC	default.AdvertisementsDB.dbo.DG ACC	N/A	Microsoft table
8	DG PC TYPE	default.AdvertisementsDB.dbo.DG PC TYPE	N/A	Microsoft table
9	GEORGE WHITES TABLE	default.AdvertisementsDB.dbo.GEORGE WHITES TABLE	N/A	Microsoft table
10	IEMC	default.AdvertisementsDB.dbo.IEMC	N/A	Microsoft table
11	ITEM ATTRIBUTES	default.AdvertisementsDB.dbo.ITEM ATTRIBUTES	N/A	Microsoft table

Information related to Data Functions calibration

It is helpful for the person who will receive the functional sizing results to know if adjustments have been done during the calibration phase. The SQL queries presented here allow you to collect this information.

Collect Data Functions that have been merged

The following SQL query collects the Data Functions that have been merged during the calibration phase:

```
set search_path=<Prefix>_local;

select * from (
select dtf.object_id as maintable_id,
cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtf.DET as DET,
dtf.RET as RET,
case dtf.isinternal when 0 then 'EIF' when 1 then 'ILF' END as DefaultType,
case dtf.user_isinternal when 0 then 'EIF' when 1 then 'ILF' END as OverwriteType,
dtf.ilf_ex as DefaultFPValue,
dtf.user_fp_value as OverwriteFPValue,
case dtf.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_datafunction dtf, cdt_objects cob
where dtf.maintable_id = cob.object_id
and dtf.cal_mergeroot_id = 0 -- not a sub transaction
and dtf.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
union all
select dtf.cal_mergeroot_id as form_id,' |-------' || cob.object_name as transaction,
cob.object_fullname as transactionfullname,
dtf.DET as DET,
dtf.RET as RET,
case dtf.isinternal when 0 then 'EIF' when 1 then 'ILF' END as DefaultType,
case dtf.user_isinternal when 0 then 'EIF' when 1 then 'ILF' END as OverwriteType,
dtf.ilf_ex as DefaultFPValue,
dtf.user_fp_value as OverwriteFPValue,
case dtf.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored'  when 258 then 'Child and Ignored' END as Status
from dss_datafunction dtf, cdt_objects cob
where dtf.maintable_id = cob.object_id
and dtf.cal_mergeroot_id > 0 -- not a sub transaction
and dtf.cal_flags not in (  8, 10, 126, 128,136, 138, 256, 258 ) -- transaction standalone or Root
) as result
order by 1 ASC, 10 DESC;
```

The result should look like this:

	form_id integer	transaction character varying	transactionfullname character varying(255)	det integer	ret integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	status text
6	11523509	IntApp ApplicationEventStart	default.LoggingDB.d	25	2	ILF		10		STD
7	11523510	IntApp Application	default.LoggingDB.d	27	2	ILF		10		STD
8	11523511	WSIRIGEP	default.ProductDB.d	37	2	ILF		10		Root
9	11523511	|-------WSIRIGEP	default.ProductDB P	37	1	EIF		5		Child
10	11523512	|-------WebOverridePricing	default.ProductDB P	2	1	EIF		5		Child
11	11523513	|-------Warranty ManufacturerRequirement D	default.ProductDB P	7	1	EIF		5		Child
12	11523514	|-------Warranty ManufacturerRequirement	default.ProductDB P	10	2	EIF		5		Child
13	11523514	|-------Warranty ManufacturerRequirement D	default.ProductDB.d	7	1	ILF		7		Child
14	11523515	Warranty Manufacturer Download	default.ProductDB.d	5	2	ILF		7		Root

Collect Data Functions that have been adjusted

The following SQL query collects the Data Functions for which the type or the Function Point value has been adjusted during the calibration phase:

```
set search_path=<Prefix>_local;

SELECT cob.object_name as LogicalFile, cob.object_fullname as LogicalFileFullname, dtf.DET as DET, dtf.RET as
RET,
case dtf.isinternal when 0 then 'EIF' when 1 then 'ILF' END as DefaultType,
case dtf.user_isinternal when 0 then 'EIF' when 1 then 'ILF' END as OverwriteType,
dtf.ilf_ex as DefaultFPValue,
dtf.user_fp_value as OverwriteFPValue
from dss_datafunction dtf, cdt_objects cob
where dtf.maintable_id = cob.object_id
and (dtf.user_fp_value >= 0 or dtf.user_isinternal >= 0)
order by 2 ASC;
```

Collect ignored or deleted Data Functions

The following SQL query collects the Data Functions that have been ignored or deleted during the calibration phase:

```
set search_path=<Prefix>_local;

SELECT cob.object_name as LogicalFile,
cob.object_fullname as LogicalFileFullname,
dtf.DET as DET,
dtf.RET as RET,
CASE dtf.isinternal when 0 then 'EIF' when 1 then 'ILF' END as DefaultType,
CASE dtf.user_isinternal when 0 then 'EIF' when 1 then 'ILF' END as OverwriteType,
dtf.ilf_ex as DefaultFPValue, dtf.user_fp_value as OverwriteFPValue,
CASE dtf.cal_flags when 0 then 'STD' when 2 then 'Root' when 4 then 'Child' when 4 then 'Child' when 8 then
'Deleted' when 10 then 'Root and Deleted' when 128 then 'Deleted' when 136 then 'Root and Deleted' when 138
then 'Child and Deleted' when 256 then 'Root and Ignored' when 258 then 'Child and Ignored' END as Status
FROM dss_datafunction dtf, cdt_objects cob
WHERE dtf.maintable_id = cob.object_id
AND dtf.cal_flags in (8, 10, 128, 136, 138, 256, 158, 160)
--AND (dtf.user_fp_value >= 0 or dtf.user_isinternal >= 0)
ORDER BY 9 ASC;
```

The result should look like this:

	logicalfile character varying(255)	logicalfilefullname character varying(255)	det integer	ret integer	defaulttype text	overwritetype text	defaultfpvalue integer	overwritefpvalue integer	cal_flags integer	status text
377	T1	default.StatisticsT	4	1	EIF		5		256	Root and Ignored
378	T0	default.StatisticsT	2	1	EIF		5		256	Root and Ignored
379	CustomersToPurge	default.CustomerDB.	1	1	EIF		5		256	Root and Ignored
380	CustomerSearchKeywo	default.CustomerDB.	3	1	EIF		5		256	Root and Ignored
381	CustomerItemPricing	default.CustomerDB.	5	1	ILF		7		256	Root and Ignored
382	Customer Maintenanc	default.CustomerDB.	16	1	EIF		5		256	Root and Ignored
383	AddressGeocode	default.CustomerDB.	18	2	EIF		5		256	Root and Ignored
384	AccountCodingSeeds	default.CustomerDB.	2	1	ILF		7		256	Root and Ignored
385	AccountClassificati	default.CustomerDB.	3	1	ILF		7		256	Root and Ignored
386	AccountClassificati	default.CustomerDB.	4	1	ILF		7		256	Root and Ignored