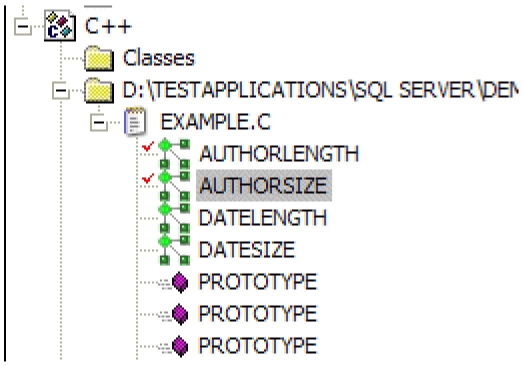
Creating new links between existing unlinked objects using CI_LINKS

Based on object's ID

Objects belong to the same project and to different projects (identical for each)

This example will explain how to create a "Call" link between unlinked C++ macros ("AUTHORLENGTH" (caller) and "AUTHORSIZE" (called)) that already exist in the CAST Analysis Service. Both objects are part of the same project (although the process for linking objects that belong to different projects is identical):



You can also check that the two objects are not already linked by executing the following query against the CAST Analysis Service:

```
select count(1) Cnt
from CTV_GUID_LINKS
where CALLER_NAME = 'AUTHORLENGTH'
and CALLED_NAME = 'AUTHORSIZE'
go
```

The result confirms that the two objects are not already linked:

```
Cnt
[int]
----
0
```

The next step is to determine the name of the link you would like to create between the two objects. In this example, a **Call** link will be created. Query the CTV_LINK_TYPES view to find out the LINK_TYPE_NAME of the link you want to create. For this example the LINK_TYPE_NAME is "callLink".

The next step is to insert the data into the CAST entry tables that will cause the new link to be created between the two C++ macros when the tool is run. Add a new **Update CAST Knowledge Base Tool** and enter the following query:

```
insert into <KB_name>.dbo.CI_LINKS (CALLER_ID, CALLED_ID, LINK_TYPE, ERROR_ID)
select
(select OBJECT_ID from <KB_name>.dbo.CTV_GUID_OBJECTS
where OBJECT_NAME='AUTHORLENGTH') CALLER_ID,
(select OBJECT_ID from <KB_name>.dbo.CTV_GUID_OBJECTS
where OBJECT_NAME='AUTHORSIZE') CALLED_ID, 'callLink', 0
go
```

- The 0 parameter will enter 0 in the ERROR_ID column of the CI_LINKS table.
- Note that you need to specify the Analysis Service database and user as above "<KB_name>.dbo" otherwise the query will fail when the job is run.
- You can also use variables to replace the Analysis Service name if required. See the on-line Help for the CAST Management Studio for more information.

Then:

- Complete the configuration of the tool and run it as outlined in What is the Update CAST Knowledge Base Tool.
- Make sure you then update the CAST System Views.

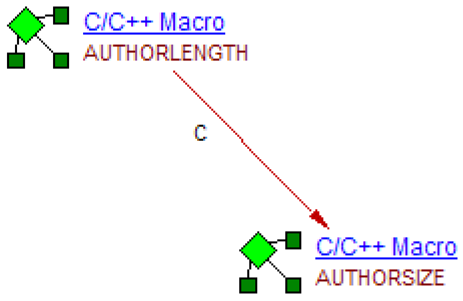
If you want to check that the new link has been created successfully, you can use the following query in an SQL IDE against your Analysis Service (adapting it for your environment):

```
select cl.LINK_TYPE_LO, cl.LINK_TYPE_HI, cl.LINK_TYPE_LO2, cl.LINK_TYPE_HI2
from CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where cgl.CALLER_NAME = 'AUTHORLENGTH'
and cgl.CALLED_NAME = 'AUTHORSIZE'
go
```

The result shows a successful link creation:

```
LINK_TYPE_LO  LINK_TYPE_HI  LINK_TYPE_LO2  LINK_TYPE_HI2
[int]         [int]         [int]          [int]
2097152       0             0              0
1 row affected by this transaction.
```

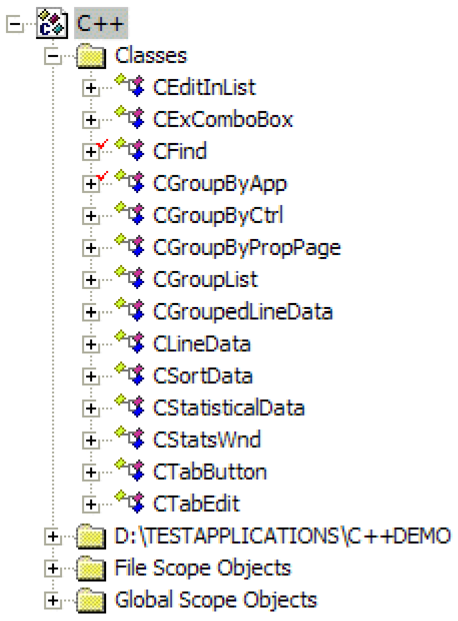
You can also check in CAST Enlighten - F5 to refresh the Object Browser - new Call link (C) between the two objects:



Based on object's GUID

Both objects have an OBJECT_GUID and belong to the same project

This example will explain how to create a "Friend" link between unlinked objects ("CFind" (caller) and "CGroupByApp" (called)) that already exist in the CAST Analysis Service. Both objects have OBJECT_GUIDs (they are C++ classes) and both are part of the same project:



The first step is to retrieve the objects' OBJECT_GUID from the CAST System Views. Use the following queries in an SQL IDE against your CAST Analysis Service:

CFind

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'CFind'
Go
```

This query returns the OBJECT_GUID for the "CFind" Class:

C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"

CGroupByApp

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'CGroupByApp'
Go
```

This query returns the OBJECT_GUID for the "CGroupByApp" Class:

```
C_Cl.CGroupByApp.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPBY..H"
```

You can also check that the two objects are not already linked by executing the following query against the CAST Analysis Service (notice we are using the objects' OBJECT_GUIDs):

```
select
count(1) Cnt
from
CTV_GUID_LINKS
where
CALLER_GUID = 'C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"'
and CALLED_GUID = 'C_Cl.CGroupByApp.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPBY..H"'
```

The result confirms that the two objects are not already linked:

```
Cnt
[int]
----
0
```

The next step is to determine the name of the link you would like to create between the two objects. In this example, a **Friend** link will be created. Query the CTV_LINK_TYPES view to find out the LINK_TYPE_NAME of the link you want to create. For this example the LINK_TYPE_NAME is "friendLink".

The next step is to insert the data into the CAST entry tables that will cause the new link to be created between the two C++ classes when the tool is run. Add a new **Update CAST Knowledge Base Tool** and enter the following query:

```
insert into <KB_name>.dbo.CI_LINKS (CALLER_GUID, CALLED_GUID, LINK_TYPE, ERROR_ID)
select 'C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"',
'C_Cl.CGroupByApp.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPBY..H"', 'friendLink', 0
go
```

- The 0 parameter will enter 0 in the ERROR_ID column of the CI_LINKS table.
- Note that you need to specify the Analysis Service database and user as above "<KB_name>.dbo" otherwise the query will fail when the job is run.
- You can also use variables to replace the Analysis Service name if required. See the on-line Help for the CAST Management Studio for more information.

Then:

- Complete the configuration of the tool and run it as outlined in What is the Update CAST Knowledge Base Tool.
- Make sure you then update the CAST System Views.

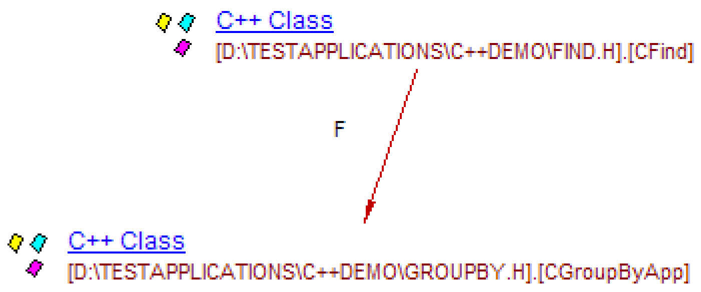
If you want to check that the new link has been created successfully, you can use the following query in an SQL IDE against your Analysis Service (adapting it for your environment):

```
select
cl.LINK_TYPE_LO, cl.LINK_TYPE_HI, cl.LINK_TYPE_LO2, cl.LINK_TYPE_HI2
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_GUID) = 'C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"'
and upper(cgl.CALLED_GUID) = 'C_Cl.CGroupByApp.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPBY..H"'
go
```

The result shows a successful link creation:

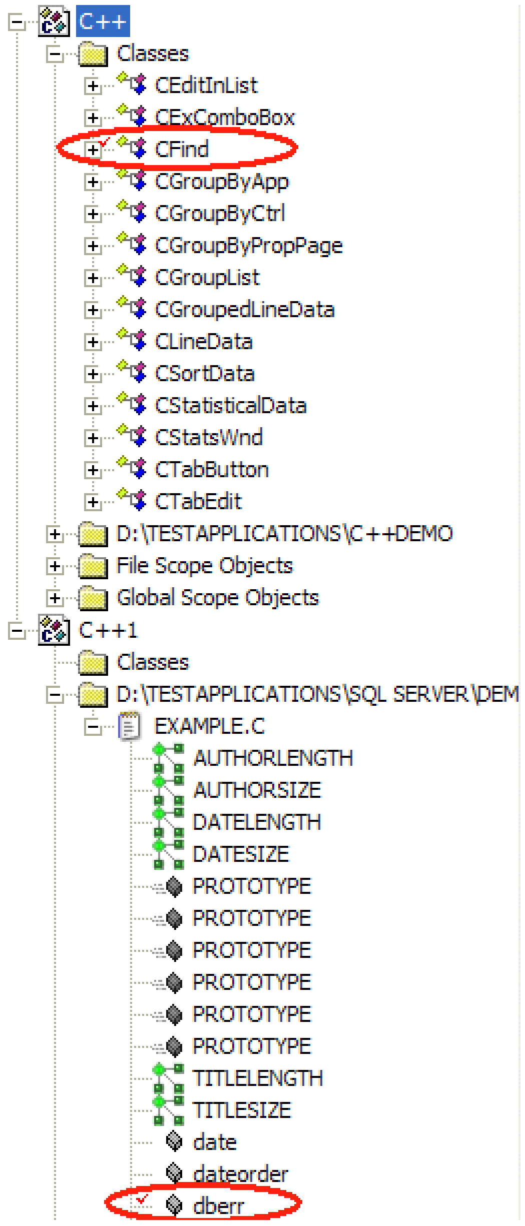
```
LINK_TYPE_LO  LINK_TYPE_HI  LINK_TYPE_LO2  LINK_TYPE_HI2
[int]         [int]         [int]          [int]
4             0             0              0
1 row affected by this transaction.
```

You can also check in CAST Enlighten - F5 to refresh the Object Browser - new Friend link (F) between the two objects:


C++ Class
[D:\TESTAPPLICATIONS\C++DEMO\FIND.H].[CFind]
F
C++ Class
[D:\TESTAPPLICATIONS\C++DEMO\GROUPBY.H].[CGroupByApp]

Both objects have an OBJECT_GUID but belong to different projects

This example will explain how to create a "Friend" link between unlinked objects ("CFind" (caller) and "dberr" (called)) that already exist in the CAST Analysis Service. Both objects have OBJECT_GUIDs (one is a C++ Class the other is a C++ Global Variable) but belong to different projects. In this case the link will belong to the "caller" project:


```
C++
  Classes
    CEditInList
    CExComboBox
    CFind
    CGroupByApp
    CGroupByCtrl
    CGroupByPropPage
    CGroupList
    CGroupedLineData
    CLineData
    CSortData
    CStatisticalData
    CStatsWnd
    CTabButton
    CTabEdit
  D:\TESTAPPLICATIONS\C++DEMO
  File Scope Objects
  Global Scope Objects
C++1
  Classes
  D:\TESTAPPLICATIONS\SQL SERVER\DEM
    EXAMPLE.C
      AUTHORLENGTH
      AUTHORSIZE
      DATELENGTH
      DATESIZE
      PROTOTYPE
      PROTOTYPE
      PROTOTYPE
      PROTOTYPE
      PROTOTYPE
      PROTOTYPE
      TITLELENGTH
      TITLESIZE
      date
      dateorder
      dberr
```

The first step is to retrieve the objects' OBJECT_GUID from the CAST System Views. Use the following queries in an SQL IDE against your CAST Analysis Service:

CFind

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'CFind'
Go
```

This query returns the OBJECT_GUID for the "CFind" Class:

```
C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"
```

dberr

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'dberr'
Go
```

This query returns the OBJECT_GUID for the "dberr" Global Variable:

```
C_GlVar.dberr.C_Fi."D:\TESTAPPLICATIONS\SQL SERVER\DEMO_C\EXAMPLE..C"
```

You can also check that the two objects are not already linked by executing the following query against the CAST Analysis Service (notice we are using the objects' OBJECT_GUIDs):

```
select
count(1) Cnt
from
CTV_GUID_LINKS
where
CALLER_GUID = 'C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"'
and CALLED_GUID = ' C_GlVar.dberr.C_Fi."D:\TESTAPPLICATIONS\SQL SERVER\DEMO_C\EXAMPLE..C"'
```

The result confirms that the two objects are not already linked:

```
Cnt
[int]
----
0
```

The next step is to determine the name of the link you would like to create between the two objects. In this example, a **Friend** link will be created. Query the CTV_LINK_TYPES view to find out the LINK_TYPE_NAME of the link you want to create. For this example the LINK_TYPE_NAME is "friendLink".
The next step is to insert the data into the CAST entry tables that will cause the new link to be created between the two C++ objects when the tool is run. Add a new **Update CAST Knowledge Base Tool** and enter the following query:

```
insert into .dbo.CI_LINKS (CALLER_GUID, CALLED_GUID, LINK_TYPE, ERROR_ID)
select
'C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"',
'C_GlVar.dberr.C_Fi."D:\TESTAPPLICATIONS\SQL SERVER\DEMO_C\EXAMPLE..C"',
'friendLink',
0
go
```

- The 0 parameter will enter 0 in the ERROR_ID column of the CI_LINKS table.
- Note that you need to specify the Analysis Service database and user as above "<KB_name>.dbo" otherwise the query will fail when the job is run.
- You can also use variables to replace the Analysis Service name if required. See the on-line Help for the CAST Management Studio for more information.

Then:

- Complete the configuration of the tool and run it as outlined in What is the Update CAST Knowledge Base Tool.
- Make sure you then update the CAST System Views.

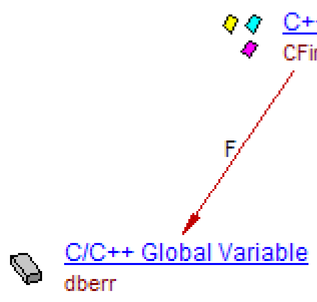
If you want to check that the new link has been created successfully, you can use the following query in an SQL IDE against your Analysis Service (adapting it for your environment):

```
select
cl.LINK_TYPE_LO,
cl.LINK_TYPE_HI,
cl.LINK_TYPE_LO2,
cl.LINK_TYPE_HI2
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_GUID) = 'C_Cl.CFind.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\FIND..H"'
and upper(cgl.CALLED_GUID) = 'C_GlVar.dberr.C_Fi."D:\TESTAPPLICATIONS\SQL SERVER\DEMO_C\EXAMPLE..C"'
go
```

The result shows a successful link creation (4 is the value used to describe the "friendLink"):

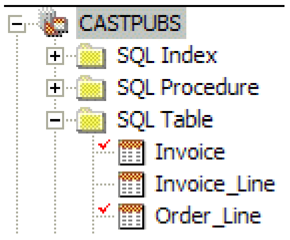
```
LINK_TYPE_LO  LINK_TYPE_HI  LINK_TYPE_LO2  LINK_TYPE_HI2
[int]         [int]         [int]          [int]
4             0             0              0
1 row affected by this transaction.
```

You can also check in CAST Enlighten - F5 to refresh the Object Browser - new Friend link (F) between the two objects:



Neither objects have an OBJECT_GUID but belong to the same project

This example will explain how to create a "Rely On" link between unlinked objects ("Invoice" (caller) and "Order_Line" (called)) that already exist in the CAST Analysis Service. Neither objects have OBJECT_GUIDs (they are T-SQL tables) but both are part of the same project:



The first step is to check that the objects do indeed have no OBJECT_GUIDs in the CAST System Views. Use the following queries in an SQL IDE against your CAST Analysis Service:

```
Invoice
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'Invoice'
Go
Order_Line
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'Order_Line'
Go
```

Both queries return NULL indicating the objects do not have an OBJECT_GUID. You can also check that the objects are not linked using the following query:

```
select
count(1) Cnt
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_NAME) = 'Invoice'
and upper(cgl.CALLED_NAME) = 'Order_Line'
go
```

This also returns a NULL result, indicating the objects are not linked.

The next step is to determine the name of the link you would like to create between the two objects. In this example, a **Rely On** link will be created. Query the CTV_LINK_TYPES view to find out the LINK_TYPE_NAME of the link you want to create. For this example the LINK_TYPE_NAME is "relyonLink".

The next step is to insert the data into the CAST entry tables that will cause the link in question to be created and also create OBJECT_GUIDs for the two objects when the tool is run. Add a new **Update CAST Knowledge Base Tool** and enter the following query:

```
insert into <KB_name>.dbo.GUID_OBJECTS (OBJECT_ID, OBJECT_GUID, ERROR_ID)
select OBJECT_ID, OBJECT_FULLNAME, 0
from <KB_name>.dbo.CTV_GUID_OBJECTS
where OBJECT_NAME = 'Invoice'
go
insert into .dbo.GUID_OBJECTS (OBJECT_ID, OBJECT_GUID, ERROR_ID)
select OBJECT_ID, OBJECT_FULLNAME, 0
from <KB_name>.dbo.CTV_GUID_OBJECTS
where OBJECT_NAME = 'Order_Line'
go
insert into <KB_name>.dbo.CI_LINKS (CALLER_GUID, CALLED_GUID, LINK_TYPE, ERROR_ID)
select
(select OBJECT_FULLNAME  from .dbo.CTV_GUID_OBJECTS where OBJECT_NAME = 'Invoice') CALLER_GUID,
(select OBJECT_FULLNAME from .dbo.CTV_GUID_OBJECTS where OBJECT_NAME = 'Order_line') CALLED_GUID,
'relyonLink',
0
go
```

This is in effect three queries:

- The first two create OBJECT_GUIDs for the two T-SQL tables using the objects' full name.
- The third inserts the data in the CI_LINKS table to create the link.

- The 0 parameter will enter 0 in the ERROR_ID column of the tables in question.
- Note that you need to specify the Analysis Service database and user as above "<KB_name>.dbo" otherwise the query will fail when the job is run.
- You can also use variables to replace the Analysis Service name if required. See the on-line Help for the CAST Management Studio for more information.
- Please see Creating OBJECT_GUIDs for more information about creating OBJECT_GUIDs for objects that do not have them.

Then:

- Complete the configuration of the tool and run it as outlined in What is the Update CAST Knowledge Base Tool.
- Make sure you then update the CAST System Views.

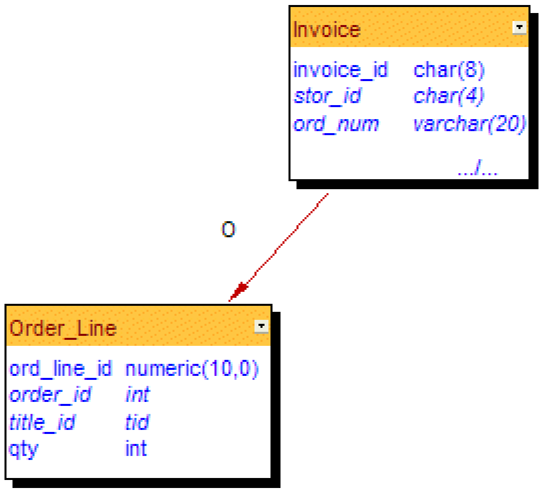
If you want to check that the new link has been created successfully, you can use the following query in an SQL IDE against your Analysis Service (adapting it for your environment):

```
select
cl.LINK_TYPE_LO,
cl.LINK_TYPE_HI,
cl.LINK_TYPE_LO2,
cl.LINK_TYPE_HI2
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_GUID) = 'WESLEY\WESLEY.CASTPUBS..Invoice'
and upper(cgl.CALLED_GUID) = 'WESLEY\WESLEY.CASTPUBS..Order_Line'
go
```

The result shows a successful link creation (32678 is the value used to describe the "relyonLink":

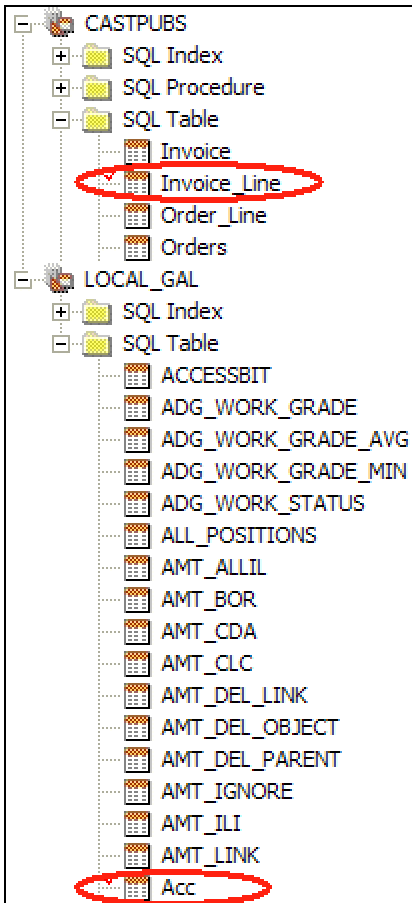
```
LINK_TYPE_LO  LINK_TYPE_HI  LINK_TYPE_LO2  LINK_TYPE_HI2
[int]         [int]         [int]          [int]
32678         0             0              0
1 row affected by this transaction.
```

You can also check in CAST Enlighten - F5 to refresh the Object Browser - new Rely On link (O) between the two objects:



Neither objects have an OBJECT_GUID and belong to different projects

This example will explain how to create a "Access" link between unlinked objects ("Invoice_Line" (caller) and "Acc" (called)) that already exist in the CAST Analysis Service. Neither object has an OBJECT_GUID (they are T-SQL tables), and both belong to different projects:



The first step is to check that the objects do indeed have no OBJECT_GUIDs in the CAST System Views. Use the following queries in an SQL IDE against your CAST Analysis Service:

Invoice_Line

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'Invoice_Line'
Go
```

Acc

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'Acc'
Go
```

Both queries return NULL indicating the objects do not have an OBJECT_GUID. You can also check that the objects are not linked using the following query:

```
select
count(1) Cnt
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_NAME) = 'Invoice_Line'
and upper(cgl.CALLED_NAME) = 'Acc'
go
```

This also returns a NULL result, indicating the objects are not linked.

The next step is to determine the name of the link you would like to create between the two objects. In this example, an **Access** link will be created. Query the CTV_LINK_TYPES view to find out the LINK_TYPE_NAME of the link you want to create. For this example the LINK_TYPE_NAME is "accessLink".

The next step is to insert the data into the CAST entry tables that will cause the link in question to be created and also create OBJECT_GUIDs for the two objects when the tool is run. Add a new **Update CAST Knowledge Base Tool** and enter the following query:

```
insert into <KB_name>.dbo.GUID_OBJECTS (OBJECT_ID, OBJECT_GUID, ERROR_ID)
select OBJECT_ID, OBJECT_FULLNAME, 0
from .dbo.CTV_GUID_OBJECTS
where OBJECT_NAME = 'Invoice_Line'
go
insert into <KB_name>.dbo.GUID_OBJECTS (OBJECT_ID, OBJECT_GUID, ERROR_ID)
select OBJECT_ID, OBJECT_FULLNAME, 0
from .dbo.CTV_GUID_OBJECTS
where OBJECT_NAME = 'Acc'
go
insert into <KB_name>.dbo.CI_LINKS (CALLER_GUID, CALLED_GUID, LINK_TYPE, ERROR_ID)
select
(select OBJECT_FULLNAME from <KB_name>.dbo.CTV_GUID_OBJECTS where OBJECT_NAME = 'Invoice_Line') CALLER_GUID,
(select OBJECT_FULLNAME from <KB_name>.dbo.CTV_GUID_OBJECTS where OBJECT_NAME = 'Acc') CALLED_GUID,
'accessLink',
0
go
```

This is in effect three queries:

- The first two create OBJECT_GUIDs for the two T-SQL tables using the objects' full name.
- The third inserts the data in the CI_LINKS table to create the link.

- The 0 parameter will enter 0 in the ERROR_ID column of the tables in question.
- Note that you need to specify the Analysis Service database and user as above "<KB_name>.dbo" otherwise the query will fail when the job is run.
- You can also use variables to replace the Analysis Service name if required. See the on-line Help for the CAST Management Studio for more information.
- Please see Creating OBJECT_GUIDs for more information about creating OBJECT_GUIDs for objects that do not have them.

Then:

- Complete the configuration of the tool and run it as outlined in What is the Update CAST Knowledge Base Tool.
- Make sure you then update the CAST System Views.

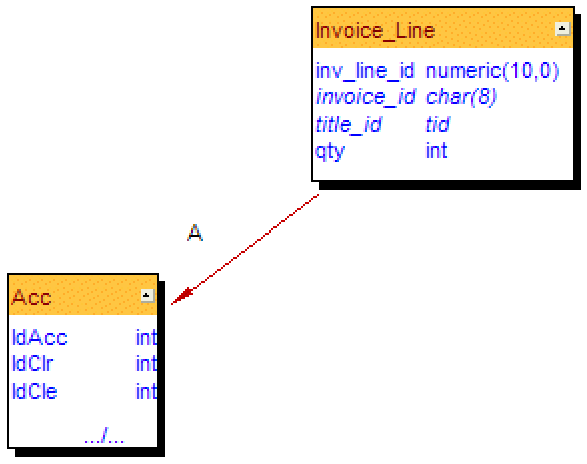
If you want to check that the new link has been created successfully, you can use the following query in an SQL IDE against your Analysis Service (adapting it for your environment):

```
select
cl.LINK_TYPE_LO,
cl.LINK_TYPE_HI,
cl.LINK_TYPE_LO2,
cl.LINK_TYPE_HI2
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_GUID) = 'WESLEY\WESLEY.CASTPUBS..Invoice_Line'
and upper(cgl.CALLED_GUID) = 'WESLEY\WESLEY.LOCAL_GAL..Acc'
go
```

The result shows a successful link creation (16777216 is the value used to describe the "accessLink"):

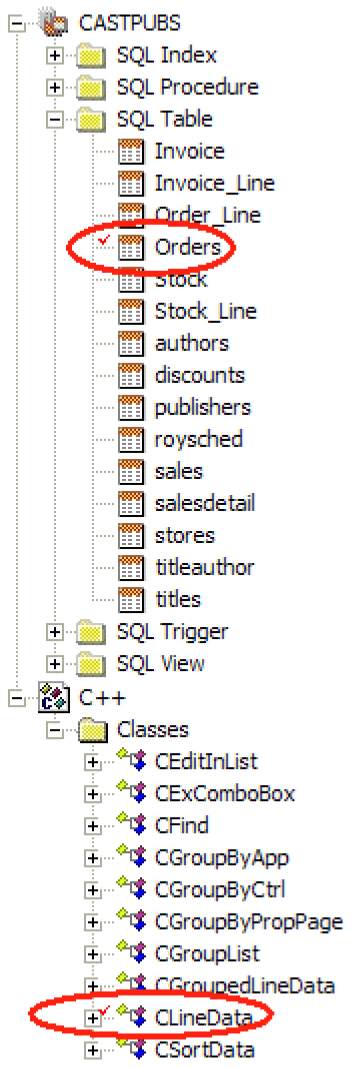
```
LINK_TYPE_LO  LINK_TYPE_HI  LINK_TYPE_LO2  LINK_TYPE_HI2
16777216      0             0              0
1 row affected by this transaction.
```

You can also check in CAST Enlighten - F5 to refresh the Object Browser - new Access link (A) between the two objects:



One object has an OBJECT_GUID and the other does not

This example will explain how to create a "Use" link between unlinked objects ("CLineData" (caller) and "Orders" (called)) that already exist in the CAST Analysis Service. The C++ class "CLineData" has an OBJECT_GUID, where as the T-SQL table "Orders" does not.



The first step is to check the objects' OBJECT_GUIDs in the CAST System Views. Use the following queries in an SQL IDE against your CAST Analysis Service:

CLineData

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'CLineData'
Go
```

Orders

```
Select OBJECT_GUID
From CTV_GUID_OBJECTS
Where OBJECT_NAME = 'Orders'
Go
```

The first query returns the following OBJECT_GUID:

```
C_Cl.CLineData.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPLIST..H"
```

The second returns NULL indicating the object does not have an OBJECT_GUID. You can also check that the objects are not linked using the following query:

```
select
count(1) Cnt
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_NAME) = 'CLineData'
and upper(cgl.CALLED_NAME) = 'Orders'
go
```

This also returns a NULL result, indicating the objects are not linked.

The next step is to determine the name of the link you would like to create between the two objects. In this example, a **Use** link will be created. Query the CTV_LINK_TYPES view to find out the LINK_TYPE_NAME of the link you want to create. For this example the LINK_TYPE_NAME is "useLink".

The next step is to insert the data into the CAST entry tables that will cause the link in question to be created and also create an OBJECT_GUID for the called object when the tool is run. Add a new **Update CAST Knowledge Base Tool** and enter the following query:

```
insert into <KB_name>.dbo.GUID_OBJECTS (OBJECT_ID, OBJECT_GUID, ERROR_ID)
select OBJECT_ID, OBJECT_FULLNAME, 0
from <KB_name>.dbo.CTV_GUID_OBJECTS
where OBJECT_NAME = 'Orders'
go
insert into <KB_name>.dbo.CI_LINKS (CALLER_GUID, CALLED_GUID, LINK_TYPE, ERROR_ID)
select
'C_Cl.CLineData.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPLIST..H"',
OBJECT_FULLNAME,
'useLink',
0
from <KB_name>.dbo.CTV_GUID_OBJECTS
where
OBJECT_NAME = 'Orders'
Go
```

This is in effect two queries:

- The first creates an OBJECT_GUID for the T-SQL table using the object's full name.
- The second inserts the data in the CI_LINKS table to create the link.

- The 0 parameter will enter 0 in the ERROR_ID column of the tables in question.
- Note that you need to specify the Analysis Service database and user as above "<KB_name>.dbo" otherwise the query will fail when the job is run.
- You can also use variables to replace the Analysis Service name if required. See the on-line Help for the CAST Management Studio for more information.
- Please see Creating OBJECT_GUIDs for more information about creating OBJECT_GUIDs for objects that do not have them.

Then:

- Complete the configuration of the tool and run it as outlined in What is the Update CAST Knowledge Base Tool.
- Make sure you then update the CAST System Views.

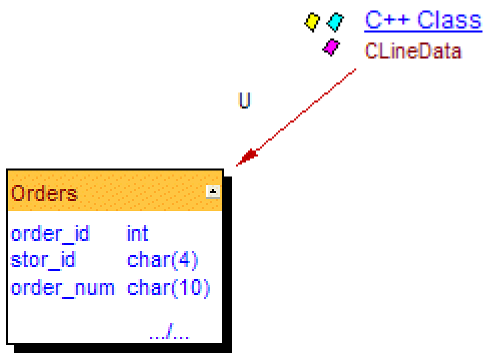
If you want to check that the new link has been created successfully, you can use the following query in an SQL IDE against your Analysis Service (adapting it for your environment):

```
select
cl.LINK_TYPE_LO,
cl.LINK_TYPE_HI,
cl.LINK_TYPE_LO2,
cl.LINK_TYPE_HI2
from
CTV_GUID_LINKS cgl join CTV_LINKS cl
on cgl.LINK_ID = cl.LINK_ID
where
upper(cgl.CALLER_GUID) = 'C_Cl.CLineData.C_Fi."D:\TESTAPPLICATIONS\C++DEMO\GROUPLIST..H"'
and upper(cgl.CALLED_GUID) = 'WESLEY\WESLEY.CASTPUBS..Orders'
go
```

The result shows a successful link creation (2097152 is the value used to describe the "useLink"):

```
LINK_TYPE_LO  LINK_TYPE_HI  LINK_TYPE_LO2 LINK_TYPE_HI2
[int]         [int]         [int]         [int]
2097152       0             0             0
1 row affected by this transaction.
```

You can also check in CAST Enlighten - F5 to refresh the Object Browser - new Access link (A) between the two objects:


C++ Class
CLineData
U
Orders
order_id int
stor_id char(4)
order_num char(10)
.../...