DataMigrator for i
User's Guide

IBM DB2 Web Query for i
DataMigrator ETL Extension

May 22, 2015

Chapter 1. Introduction

Product Description

IBM DB2 Web Query for i DataMigrator ETL Extension (DataMigrator for i) is a member of the *IBM DB2 Web Query for i* product family. It supports the building, population and maintenance of data base tables from one or more data sources. DataMigrator for i provides an Extract, Transform and Load (ETL) solution that runs on IBM i. It is useful for any IBM i customer interested in consolidating data for reporting and analytics, or for creating a more optimal reporting environment that does not interfere with an existing production environment. Customers can organize and consolidate data into a single environment that is optimized for Query Reporting, BI (Business Intelligence) and Analytics.

DataMigrator for i is **ordered** as product **5733WQM** from IBM. It is an add-on product for either edition of DB2 Web Query for i, V2.1: Express or Standard. DB2 Web Query Express (5733-WQE) or Standard (5733-WQS) is required as a prerequisite product before ordering DataMigrator for i.

> **Note:** While ordering is done for product 5733WQM, the actual DataMigrator software is **packaged** and distributed with DB2 Web Query for i product 5733WQX. It is enabled using (previously reserved) option 8 (#5108) of 5733WQX.

Product Overview

DataMigrator for i is a set of software components that automates the process of Extract, Transform and Load (ETL) of data. DataMigrator provides the capability to:

- Access data, called data sources, from database files and flat (IFS) files
- Integrate multiple data sources into a set of target tables such as a data warehouse
- Apply transformation logic to 'clean up' and convert data into a desirable types
- Aggregate data to simplify reporting and decision support
- Schedule updates to synchronize data sources

DataMigrator for i has two main components:

- The **Data Management Console (DMC)** graphical user interface is used to define data transport and transformation into a data flow through a series of drag and drop operations. The interface lets you see visualize how the data will flow and transform from the source to the target. The DMC is also used for scheduling flows, defining email notification of flow outcomes, and view logs and report statistics. The DMC is a Windows™ application.
- The **DataMigrator Server** stores and runs data flows. It also processes defined schedules. Because of its integration with DB2 Web Query, the DataMigrator Server runs in the same instance as the Web Query server on the IBM i.

The following picture depicts the overall architecture of DataMigrator for i.

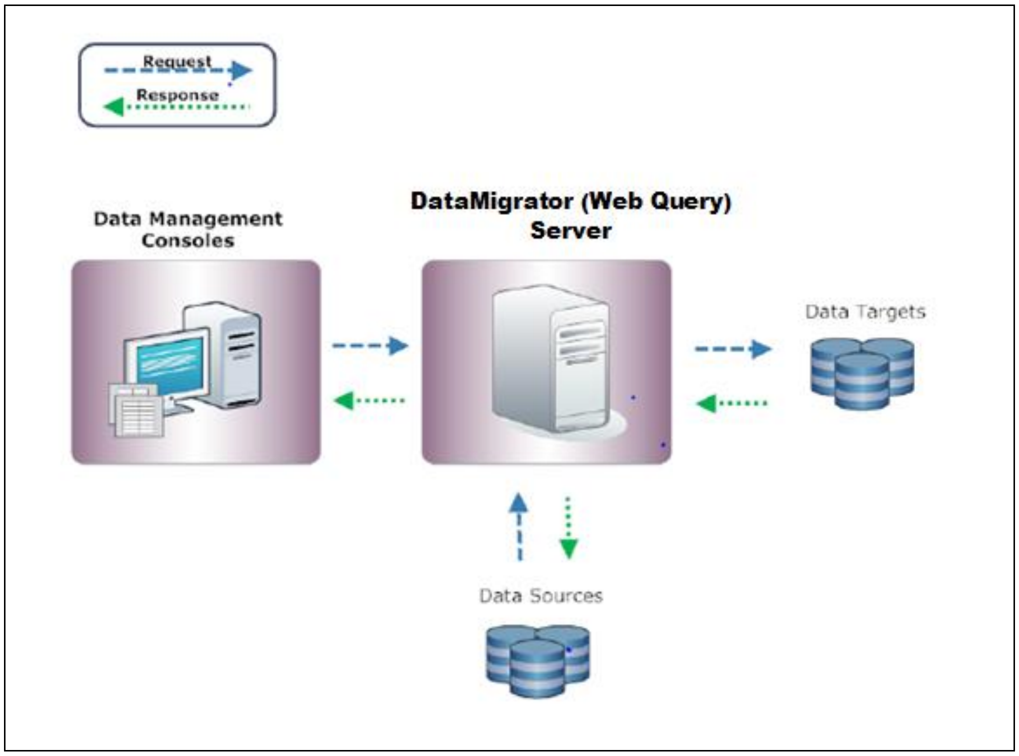


Figure 1: DataMigrator for i architecture

Through the DMC, data transport and transformation can be defined through a series of drag and drop operations. The DMC interface lets you see visualize how the data will flow and transform from the source to the target.

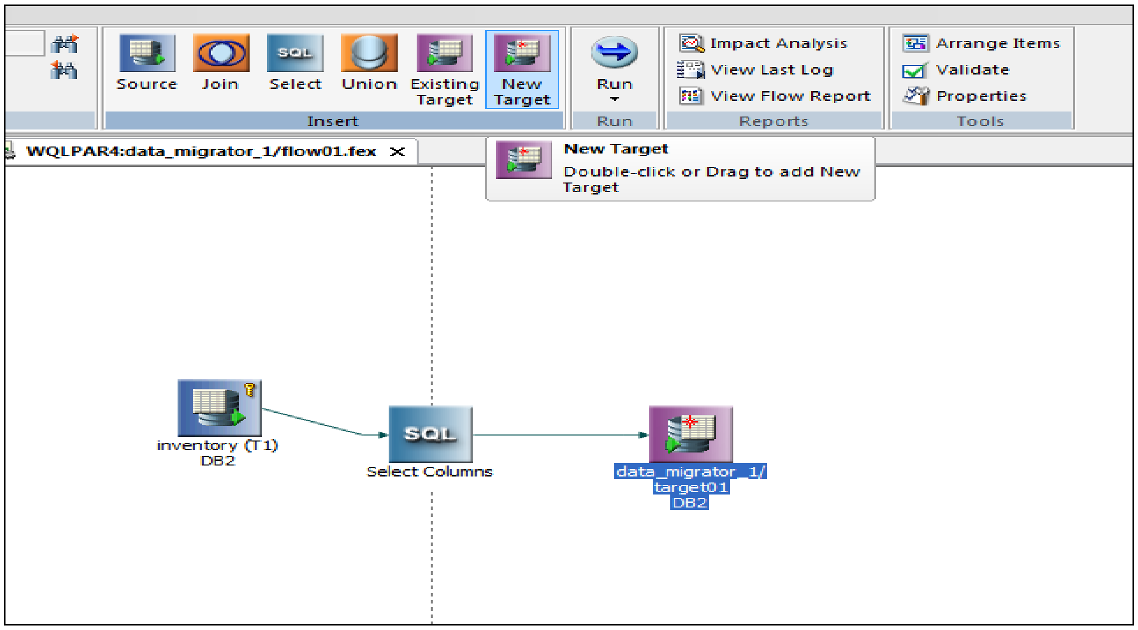


Figure 0: Graphical view of a data flow

DataMigrator for i provides the ability to perform both bulk loads of data as well as incremental updates of data. This makes it ideal for populating and maintaining a reporting database such as a data mart or data warehouse.

Defining the ETL process is done through the creation of a **Data Flow**. A Data Flow defines where the data comes from, how it should be transformed, and into what files the resulting data should be loaded. After it is defined, a data flow can be run immediately or scheduled to run once or on a recurring basis. As many data flows can be created as necessary to support an environment.

Every Data Flow is driven by a **Process Flow**. A Process Flow controls how a Data Flow runs. Every Data Flow automatically has an associated **Process Flow** created by DataMigrator for i. When you 'run' a Data Flow, you are actually running the associated Process Flow, which in turn drives the Data Flow. You can also create explicit Process Flows. This is useful, for example, when you want to run multiple data flows at once.

Every Data Flow has at least one **Data Source** and at least one **Data Target**. If you are familiar with DB2 Web Query, a Data Source and Data Target are Synonyms (Metadata) representing actual data files.

A **Data Target** is a database file. Simply put, it is the target repository for the collected data. In most cases the Data Target is one or more database tables residing on the same system on which DataMigrator for i is installed and running.

Data Sources can be any data available to DB2 Web Query for i. In fact, existing synonyms defined for DB2 Web Query for i can be used by DataMigrator for i. Database files/tables, the most common data sources, can be from the same system on which DataMigrator for i is installed, from one or more remote systems, or both. DataMigrator can also retrieve data from DB2/LUW and DB2/z Systems. Flat files in the IFS (Integrated File System) can also be used as a source of data, including using a file listener capability to help automate the process. As a unique integration point, DataMigrator can also read from journals, including remote journals, as a data source. Journals are particularly useful for incremental maintenance flows where data changes to files/tables can be captured in a very low overhead manner.

When DataMigrator is installed with DB2 Web Query Standard Edition, a Microsoft SQL Server database can also be used as a data source for the ETL process. For Oracle JDEdwards World and EnterpriseOne customers, the JDE adapter can be added to Standard Edition and subsequently used as a data source for DataMigrator.

The main interface for working with DataMigrator for i Data Flows and Process Flows is through the **Data Management Console (DMC)**, a component within DB2 Web Query Developer Workbench.

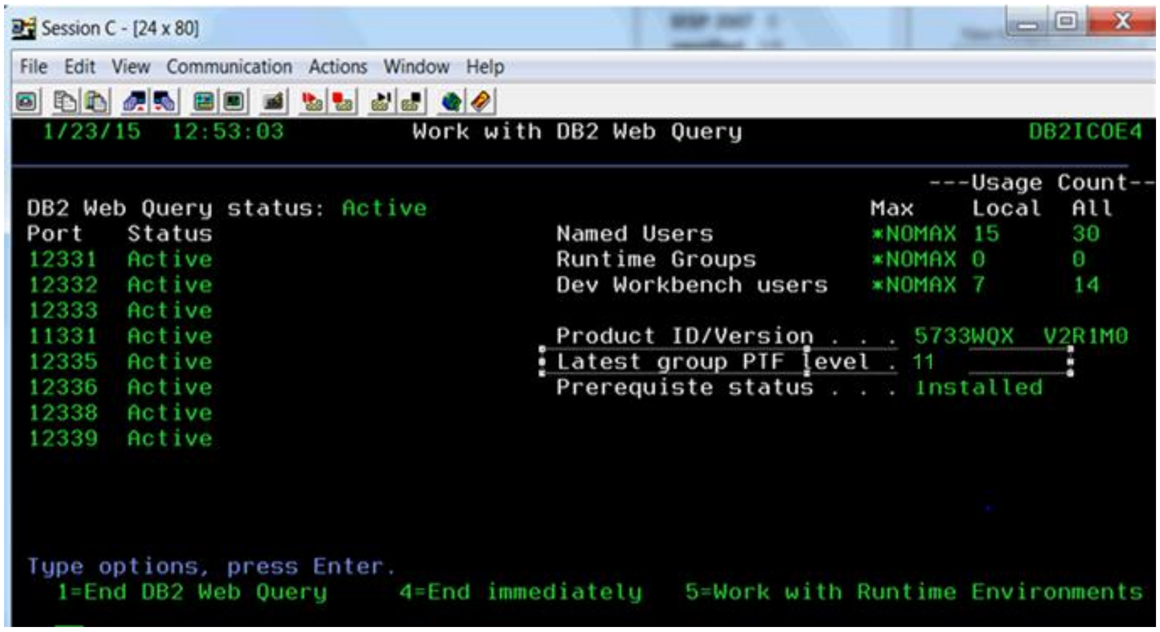
The latest information for DataMigrator for i can always be found at the DB2 Web Query for i wiki at: http://www.ibm.com/developerworks/spaces/DB2WebQuery.

Chapter 2. Setup and Configuration

Setup

Prereq: DB2 Web Query for i

DataMigrator for i software is packaged as option 8 of the DB2 Web Query for i product 5733WQX, starting with V2.1.0, HF11 (group PTF Level 11) or V2.1.1. Ensure 5733WQX is installed and runs cleanly. Ensure the latest PTF group level is installed. The PTF group are: SF99646, SF99647 or SF99747 for IBM i 6.1, 7.1 or 7.2, respectively.



```
 1/23/15  12:53:03          Work with DB2 Web Query                    DB2ICOE4

                                                              ---Usage Count--
 DB2 Web Query status: Active                          Max     Local   All
 Port    Status                  Named Users           *NOMAX  15      30
 12331   Active                  Runtime Groups        *NOMAX  0       0
 12332   Active                  Dev Workbench users   *NOMAX  7       14
 12333   Active
 11331   Active                  Product ID/Version . . .  5733WQX  V2R1M0
 12335   Active                  Latest group PTF level .  11
 12336   Active                  Prerequisite status . . . Installed
 12338   Active
 12339   Active




 Type options, press Enter.
   1=End DB2 Web Query     4=End immediately   5=Work with Runtime Environments
```

Figure 1: WRKWEBQRY command showing Hot Fix 11

Install Option 8

Install the DataMigrator for i option 8 feature using the CL command:

RSTLICPGM LICPGM(5733WQX) DEV(OPT01) OPTION(8)

For OPT01 substitute the system's optical device or whatever device where you have loaded the 5733WQX product install package you received from IBM.

Install DM License Key

If you purchased DataMigrator for i, you received a license key for option 8 from IBM. Install that key at this time (using ADDLICKEY). If you have not purchased

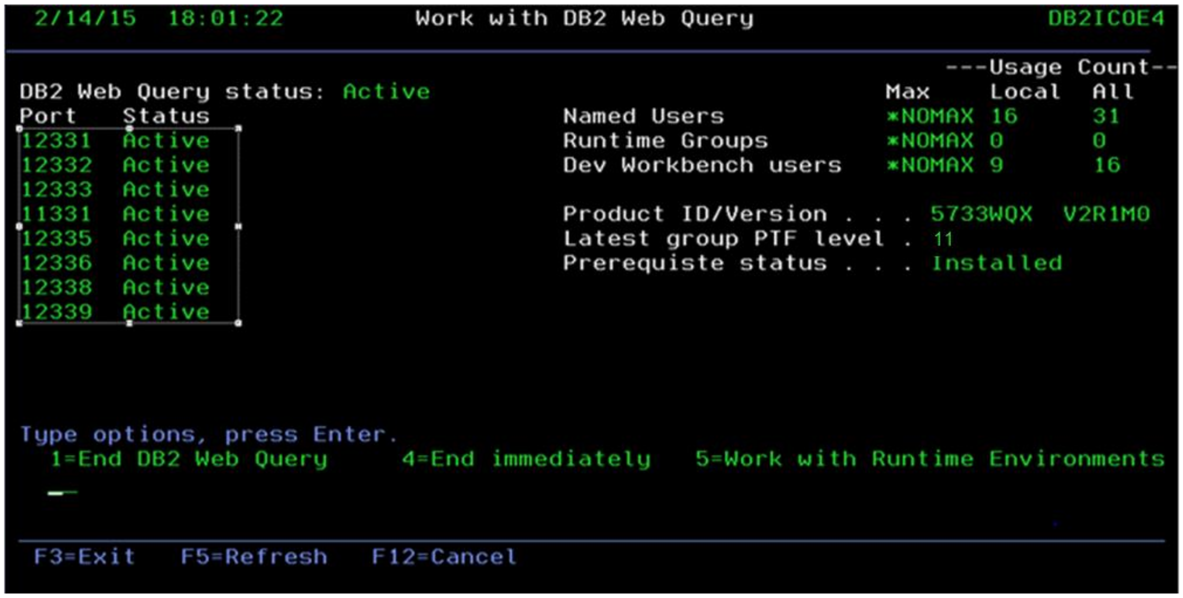
DataMigrator for i, the product will be started into a 70 day trial when you first access the server using the DMC.

Restart Web Query

End and restart DB2 Web Query for i using the CL commands:

ENDWEBQRY
STRWEBQRY

Wait for Web Query to start. Using the WRKWEBQRY CL command, wait for all ports to become active:



```
 2/14/15   18:01:22          Work with DB2 Web Query                  DB2ICOE4

                                                         ---Usage Count--
DB2 Web Query status: Active                          Max     Local   All
Port    Status                         Named Users          *NOMAX  16      31
12331   Active                         Runtime Groups       *NOMAX  0       0
12332   Active                         Dev Workbench users  *NOMAX  9       16
12333   Active
11331   Active                         Product ID/Version . . .  5733WQX   V2R1M0
12335   Active                         Latest group PTF level .  11
12336   Active                         Prerequiste status . . .  Installed
12338   Active
12339   Active



Type options, press Enter.
  1=End DB2 Web Query     4=End immediately    5=Work with Runtime Environments


 F3=Exit     F5=Refresh     F12=Cancel
```

Figure 2: WRKWEBQRY screen showing all ports 12331-1233x as Active

Server Configuration

Overview

DataMigrator for i is integrated with DB2 Web Query for i. Therefore many of the general management functions are controlled through the Web Query interfaces. These include the WRKWEBQRY CL command and the BI Portal web browser interface. The DataMigrator server component runs in the same backend server environment as Web Query. Consequently, the WRKWEBQRY, ENDWEBQRY and STRWEBQRY CL commands control the starting, ending and overall configuration of that server.

The main interface for working with DataMigrator for i Data Flows and Process Flows is through the Data Management Console (DMC), a component within DB2 Web Query Developer Workbench. We will discuss the DMC later.

DataMigrator Flow objects are managed within Web Query folders, just like Web Query reports and synonyms. **Top Level folders used for DataMigrator should be created using the BI Portal or Developer Workbench. Do not use the DMC to create top level folders.**

> Note: the BI Portal is the 12331 port interface. Access it using a web browser pointed to address: *http://<yoursystem>:12331/webquery* where <yoursystem> is the IBM i where Web Query for i is installed.

Create a Top Level Folder

Using the BI Portal, right click on highest level of the navigation tree and create a Web Query top level folder that will be used to contain DataMigrator flows.

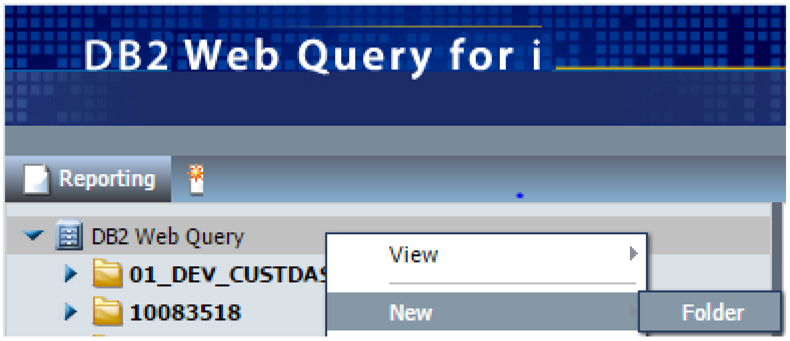
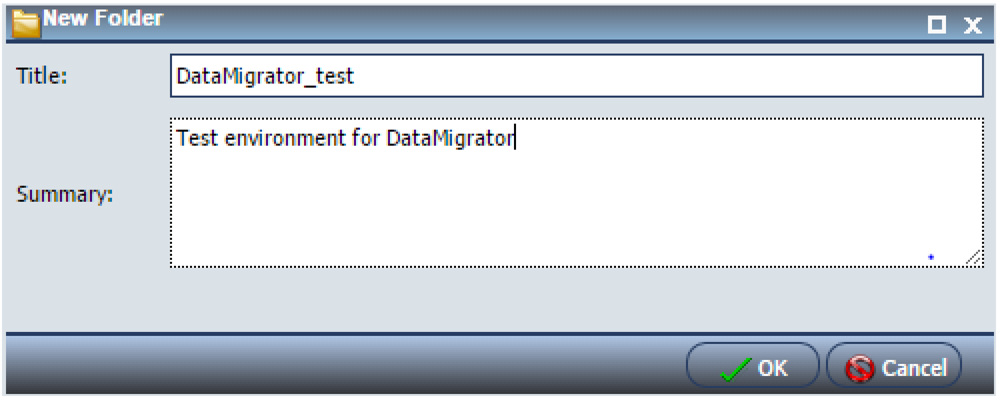


Figure 2: create new Folder on BI Portal



DataMigrator_test

Test environment for DataMigrator

Figure 3: BI Portal – New Folder

<u>DataMigrator Developer User</u>

From a user management perspective a DataMigrator developer user **is** a Developer Workbench user in Web Query. Any Developer Workbench user in Web Query is automatically allowed to be a DataMigrator user and use the DMC.

To enable someone to use DataMigrator, use the *Security Center* option under the *Administration* drop down in the BI Portal.

Note: you must be a Web Query administrator in order to access Security Center.

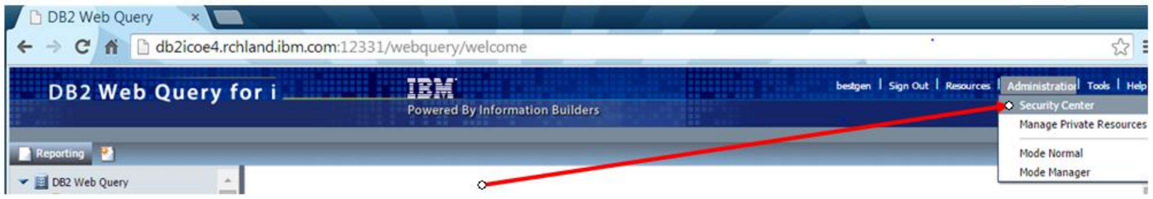
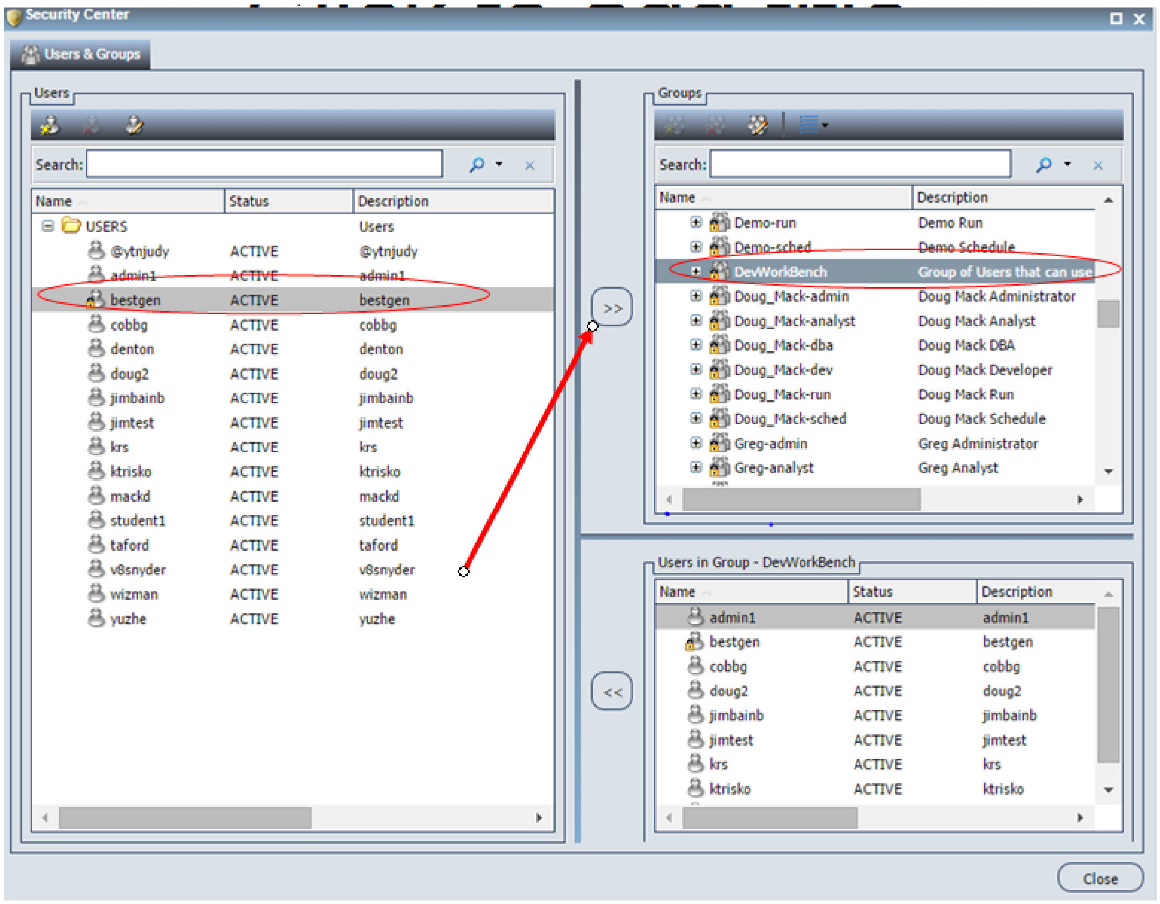


Figure 4. Security Center option in the BI Portal

Highlight the *DevWorkBench* Group and the Web Query user to be enabled for DMC and press the >> button. The user should be added to the User list in the lower right box.



Name	Status	Description
USERS		Users
@ytnjudy	ACTIVE	@ytnjudy
admin1	ACTIVE	admin1
bestgen	ACTIVE	bestgen
cobbg	ACTIVE	cobbg
denton	ACTIVE	denton
doug2	ACTIVE	doug2
jimbainb	ACTIVE	jimbainb
jimtest	ACTIVE	jimtest
krs	ACTIVE	krs
ktrisko	ACTIVE	ktrisko
mackd	ACTIVE	mackd
student1	ACTIVE	student1
taford	ACTIVE	taford
v8snyder	ACTIVE	v8snyder
wizman	ACTIVE	wizman
yuzhe	ACTIVE	yuzhe

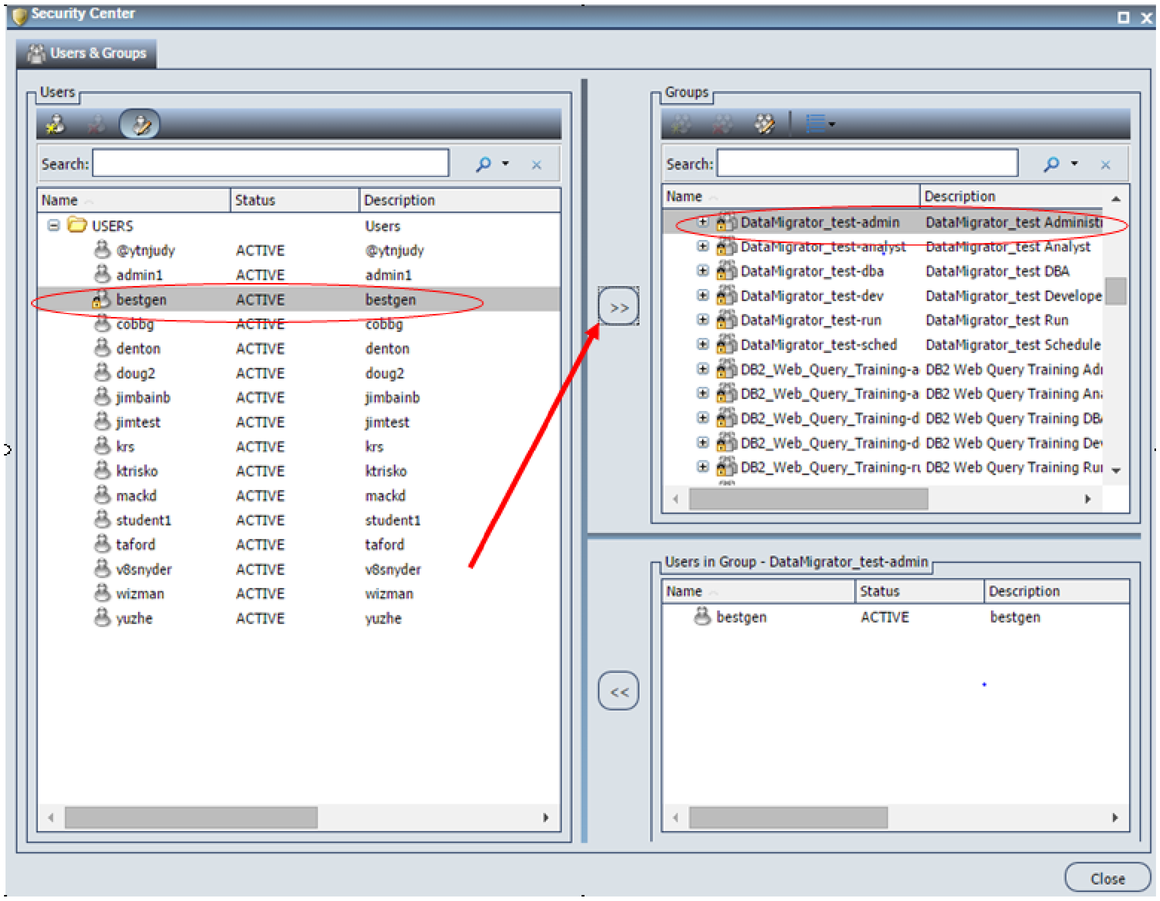
Name	Description
Demo-run	Demo Run
Demo-sched	Demo Schedule
DevWorkBench	Group of Users that can use
Doug_Mack-admin	Doug Mack Administrator
Doug_Mack-analyst	Doug Mack Analyst
Doug_Mack-dba	Doug Mack DBA
Doug_Mack-dev	Doug Mack Developer
Doug_Mack-run	Doug Mack Run
Doug_Mack-sched	Doug Mack Schedule
Greg-admin	Greg Administrator
Greg-analyst	Greg Analyst

Users in Group - DevWorkBench

Name	Status	Description
admin1	ACTIVE	admin1
bestgen	ACTIVE	bestgen
cobbg	ACTIVE	cobbg
doug2	ACTIVE	doug2
jimbainb	ACTIVE	jimbainb
jimtest	ACTIVE	jimtest
krs	ACTIVE	krs
ktrisko	ACTIVE	ktrisko

Figure 5: Enabling a user for Developer Workbench and DMC

Next, provide the user developer or admin access to the DataMigrator top level folder created earlier.



Users

Name	Status	Description
USERS		Users
@ytnjudy	ACTIVE	@ytnjudy
admin1	ACTIVE	admin1
bestgen	ACTIVE	bestgen
cobbg	ACTIVE	cobbg
denton	ACTIVE	denton
doug2	ACTIVE	doug2
jimbainb	ACTIVE	jimbainb
jimtest	ACTIVE	jimtest
krs	ACTIVE	krs
ktrisko	ACTIVE	ktrisko
mackd	ACTIVE	mackd
student1	ACTIVE	student1
taford	ACTIVE	taford
v8snyder	ACTIVE	v8snyder
wizman	ACTIVE	wizman
yuzhe	ACTIVE	yuzhe

Groups

Name	Description
DataMigrator_test-admin	DataMigrator_test Administ
DataMigrator_test-analyst	DataMigrator_test Analyst
DataMigrator_test-dba	DataMigrator_test DBA
DataMigrator_test-dev	DataMigrator_test Develope
DataMigrator_test-run	DataMigrator_test Run
DataMigrator_test-sched	DataMigrator_test Schedule
DB2_Web_Query_Training-a	DB2 Web Query Training Adı
DB2_Web_Query_Training-a	DB2 Web Query Training Ana
DB2_Web_Query_Training-d	DB2 Web Query Training DB
DB2_Web_Query_Training-d	DB2 Web Query Training Dev
DB2_Web_Query_Training-ru	DB2 Web Query Training Rur

Users in Group - DataMigrator_test-admin

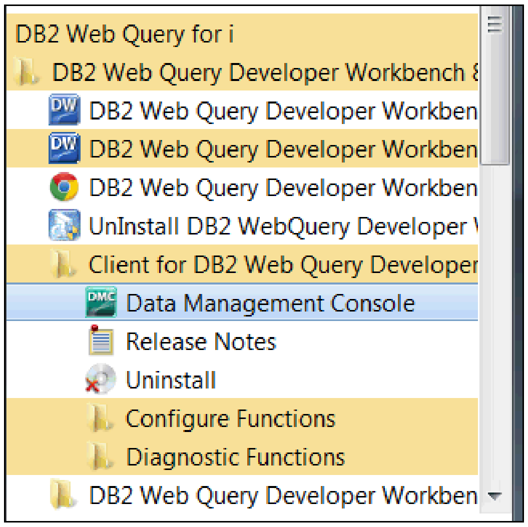
Name	Status	Description
bestgen	ACTIVE	bestgen

Figure 6: user access to the created DataMigrator folder

Install and Configure the DMC

Download the Developer Workbench client to your (the user's) PC. The Data Management Console (DMC) is part of the WorkBench install image. Download the latest Workbench client install .exe and .rar files from the IBM i server where Web Query is installed at the IFS location: /QIBM/ProdData/QWEBQRY/DeveloperWorkbench

Run the .exe file to install Workbench. This will also install the DMC. Once the install completes, find the DMC as the Data Management Console in the Windows Start menu directory.


DB2 Web Query for i
DB2 Web Query Developer Workbench 8
DB2 Web Query Developer Workben
DB2 Web Query Developer Workben
DB2 Web Query Developer Workben
UnInstall DB2 WebQuery Developer \
Client for DB2 Web Query Developer
Data Management Console
Release Notes
Uninstall
Configure Functions
Diagnostic Functions
DB2 Web Query Developer Workben
Figure 7: Find and Start DMC

The DMC window should now be showing. At this point the Web Query/DataMigrator server needs to be registered to the DMC. DataMigrator runs off the 12333 and 12332 ports. These ports are specified during the server configuration.

To add a server, highlight the Servers folder and either click the New button at the top or right click on the *Servers* folder and *Add Server Node*.

- Enter the network name of the Web Query server.
- Specify port 12333 for the HTTP Listener Port and 12332 for the TCP Listener Port.
- Security Type should be explicit. You may either provide the User ID and Password at this point or wait and be prompted during the server connection.

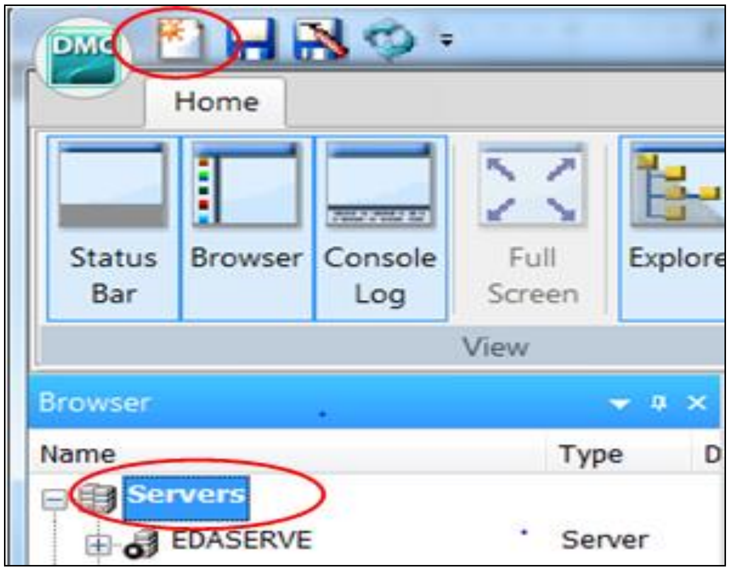
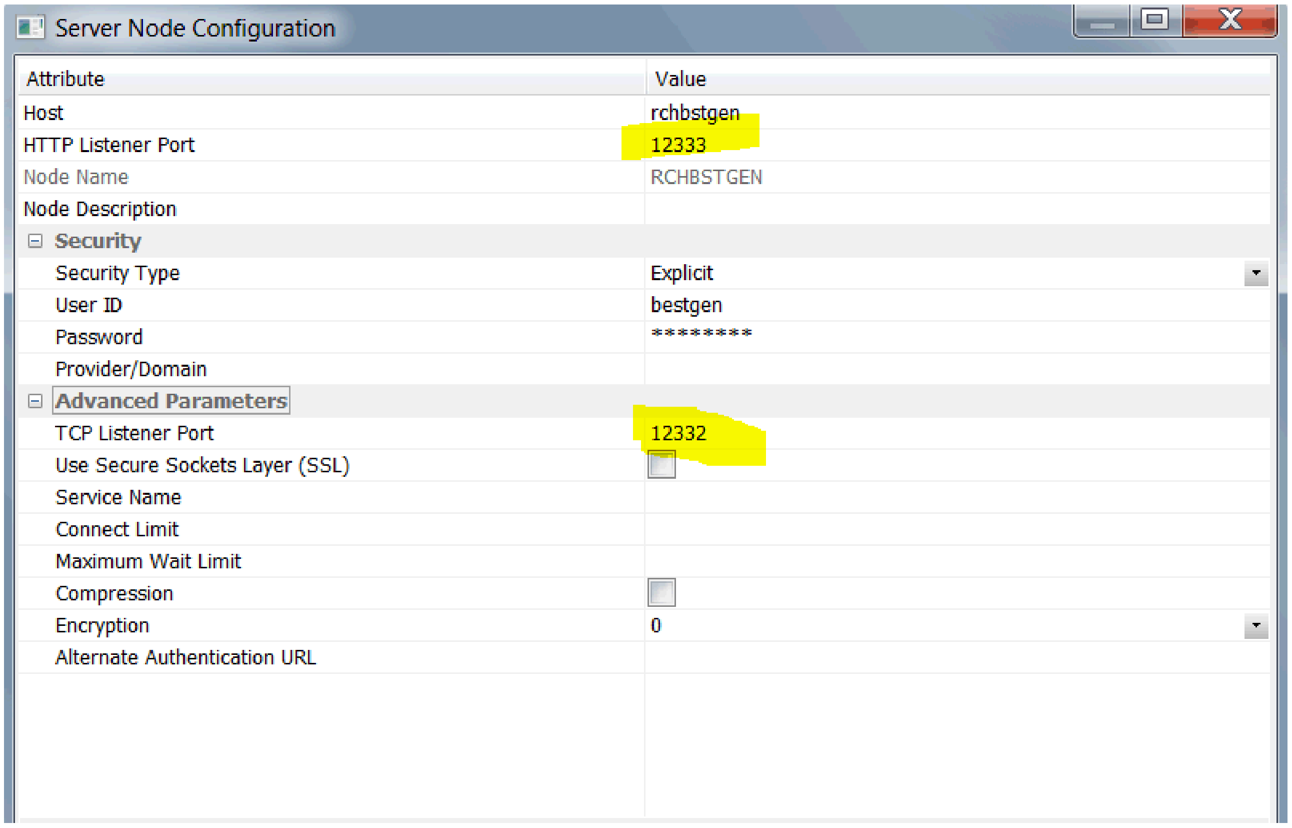


Figure 8: Add a Server



Attribute	Value
Host	rchbstgen
HTTP Listener Port	12333
Node Name	RCHBSTGEN
Node Description	
⊟ Security	
Security Type	Explicit
User ID	bestgen
Password	********
Provider/Domain	
⊟ Advanced Parameters	
TCP Listener Port	12332
Use Secure Sockets Layer (SSL)	☐
Service Name	
Connect Limit	
Maximum Wait Limit	
Compression	☐
Encryption	0
Alternate Authentication URL	

Figure 9: Configure Server connection under DMC

Click *Save.*

As the final step, register the top level folder created earlier through the BI Portal.

Find and expand the newly added server in the left navigation tree in DMC. Right click on the *Application Directories* folder under the expanded server view and choose *Application Path*.

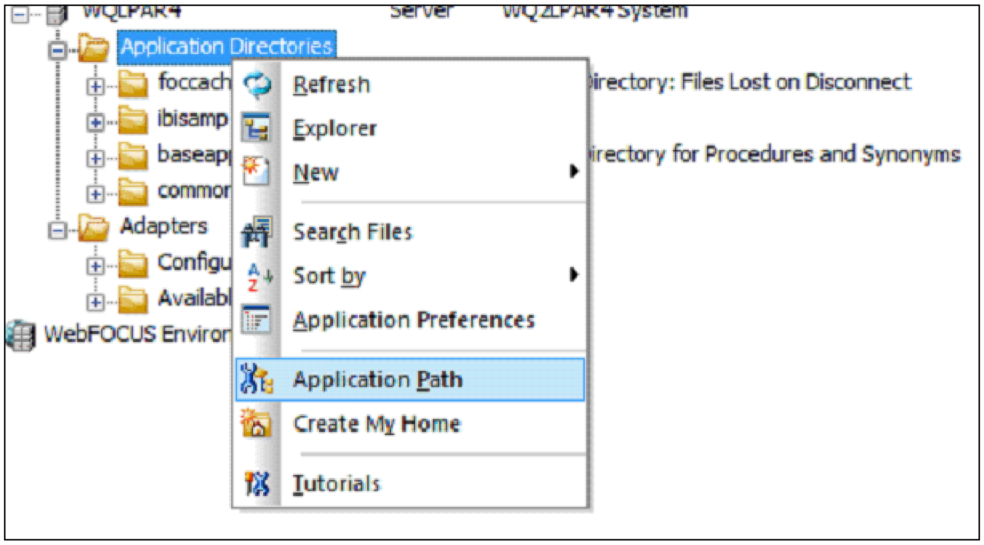


Figure 10: Choose Application Path

Click the dropdown arrow and choose *Append to previously executed profiles*.

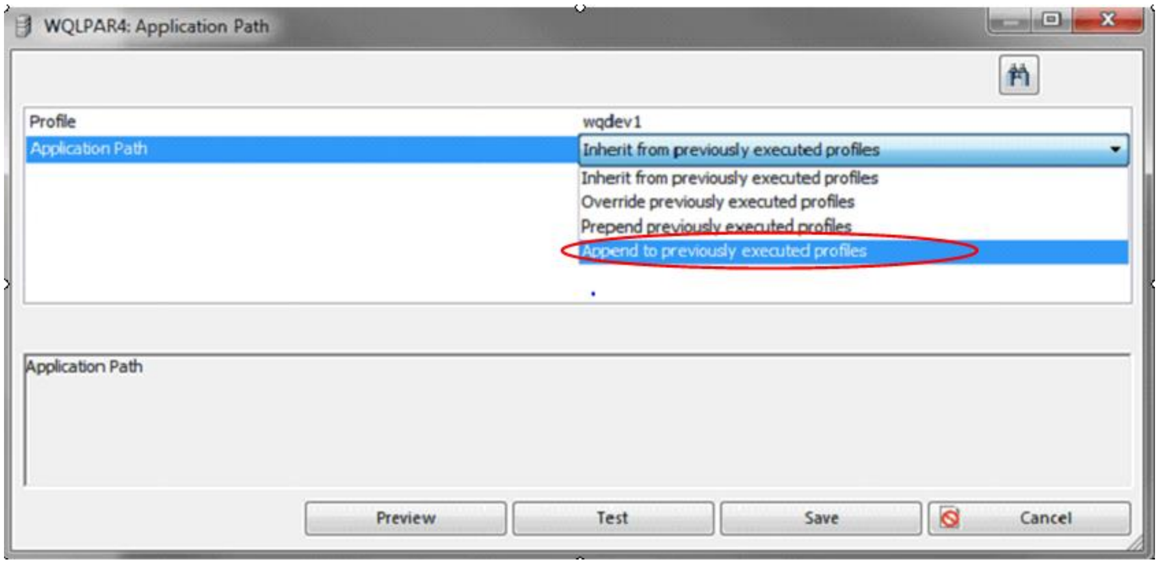
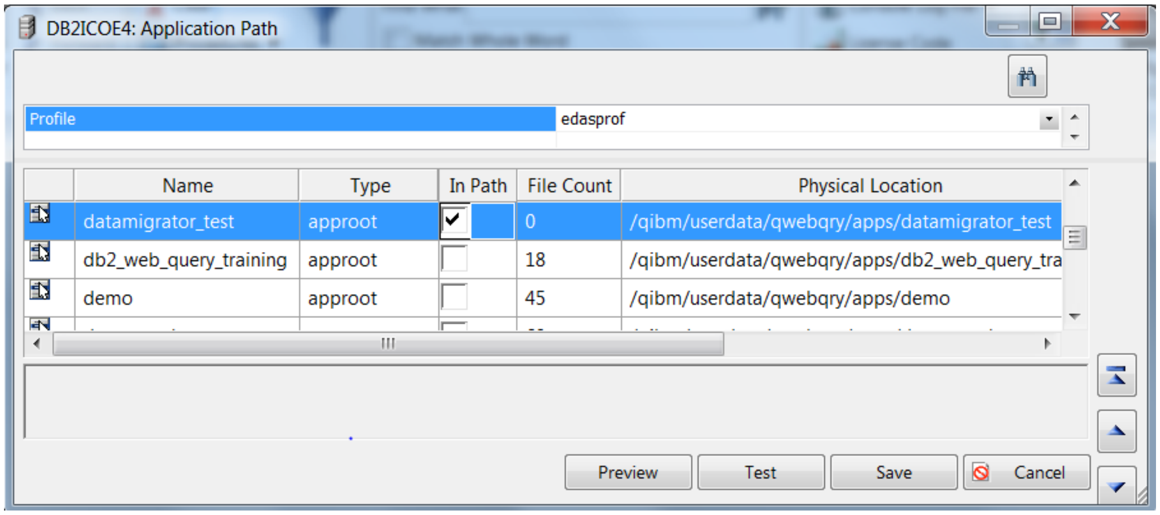

WQLPAR4: Application Path
Profile
wqdev1
Application Path
Inherit from previously executed profiles
Inherit from previously executed profiles
Override previously executed profiles
Prepend previously executed profiles
Append to previously executed profiles
Application Path
Preview
Test
Save
Cancel

Figure 11: Append to previously executed profiles

Find and select the top level folder.



DB2ICOE4: Application Path

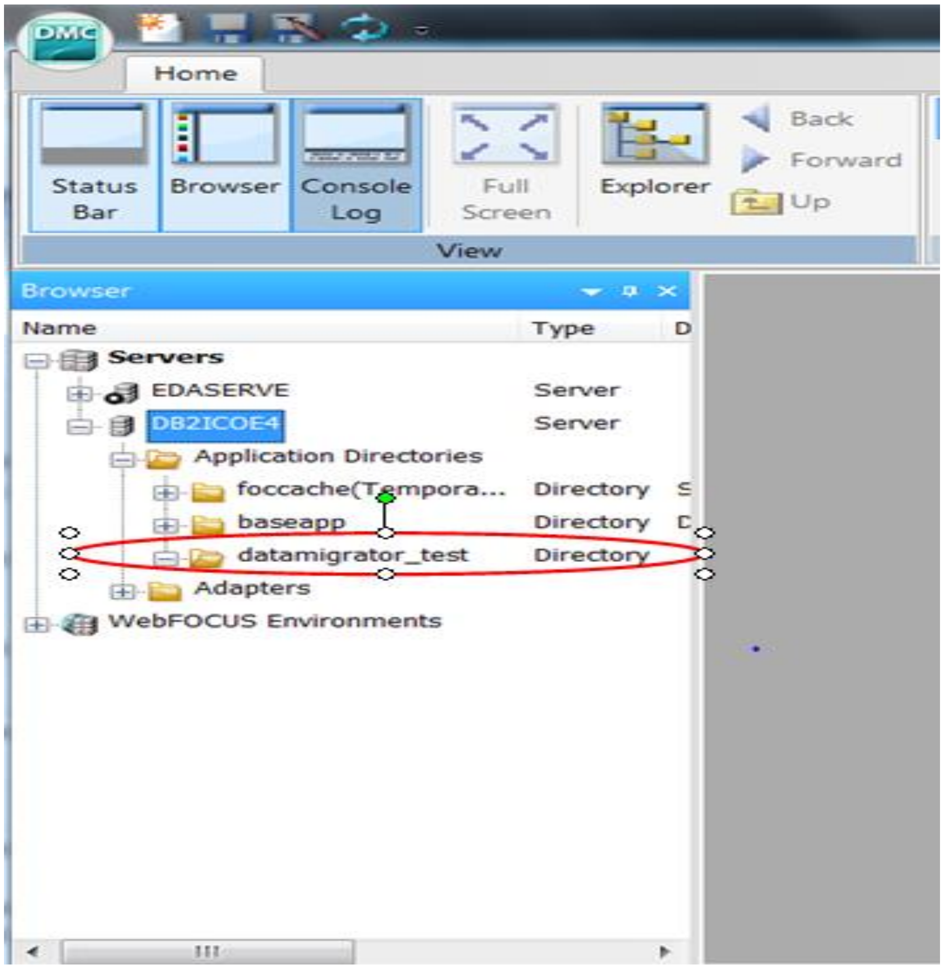
Profile	edasprof

	Name	Type	In Path	File Count	Physical Location
	datamigrator_test	approot	✓	0	/qibm/userdata/qwebqry/apps/datamigrator_test
	db2_web_query_training	approot		18	/qibm/userdata/qwebqry/apps/db2_web_query_tra
	demo	approot		45	/qibm/userdata/qwebqry/apps/demo

Figure 12: Select top level folder

Click *Save.*

At this point the top level folder should now show in the navigation tree.



Name	Type	D
Servers		
EDASERVE	Server	
DB2ICOE4	Server	
Application Directories		
foccache(Tempora...	Directory	S
baseapp	Directory	D
datamigrator_test	Directory	
Adapters		
WebFOCUS Environments		

Figure 13: top level folder in navigation tree

Note: folder *baseapp* is checked by default when adding folders to the *Application Path*. If you wish to remove it, for example to 'clean up' the list of folders under the *Application Directories* list, you may. To do so, simply choose *Application Path* again and uncheck baseapp. For this user's guide baseapp was on the list.

Verify the QWQCENT library

The examples in this user's guide use tables in the QWQCENT library. This library can also be referenced as the *Century* library. The QWQCENT library is shipped with the 5733WQX product.

The IBM DB2 Web Query development team will make occasional changes and updates to the QWQCENT library. When this happens, a new save file of the library will be shipped as part of the group PTF. The save file will always be named QWEBQRY/QWQCENT.

This user's guide uses the version of QWQCENT as it was defined as of v2.1.0 Hot Fix 11. If you have not made any customization of QWQCENT, it is highly suggested that, before you begin, you restore the latest version of the library from the shipped save file using this command:

```
RSTLIB SAVLIB(QWQCENT) DEV(*SAVF) SAVF(QWEBQRY/QWQCENT)
MBROPT(*ALL) ALWOBJDIF(*ALL)
```

If you already have a version of QWQCENT installed on your system, it is possible that certain table columns that are missing or have different names than those used in this user's guide.

Conclusion

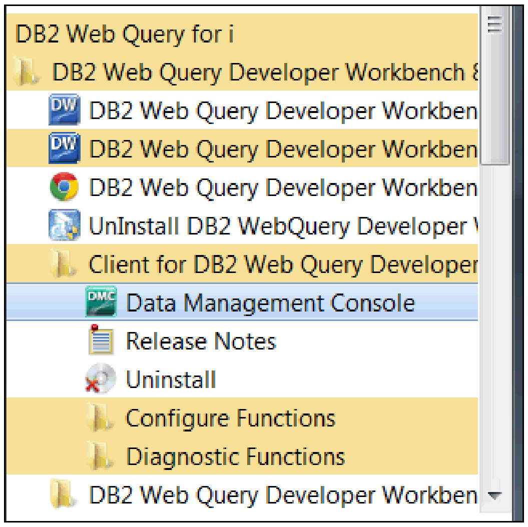
At this point setup and configuration is complete. You are ready to start creating Flows!

Chapter 3. Creating A Simple Data Flow

Overview

The Data Management Console (DMC) is the primary interface for DataMigrator for i. We will use the DMC throughout this document for building, editing and running flows.

Find and start up DMC.



DB2 Web Query for i
DB2 Web Query Developer Workbench
DB2 Web Query Developer Workben
DB2 Web Query Developer Workben
DB2 Web Query Developer Workben
UnInstall DB2 WebQuery Developer
Client for DB2 Web Query Developer
Data Management Console
Release Notes
Uninstall
Configure Functions
Diagnostic Functions
DB2 Web Query Developer Workben

Figure 3: Find and start DMC

You should already have your DataMigrator server added. Expand on the server in the navigation tree and find the *DataMigrator_test* folder made available as part of the setup.

The main development task with DataMigrator for i is creating Data Flows. As mentioned in the introduction, Data Flows define how data is gathered, processed and stored in as **ETL** process.

Before creating a data flow, let’s take a quick look at some basic DMC feature landmarks that will be useful as we use it to create and run flows.

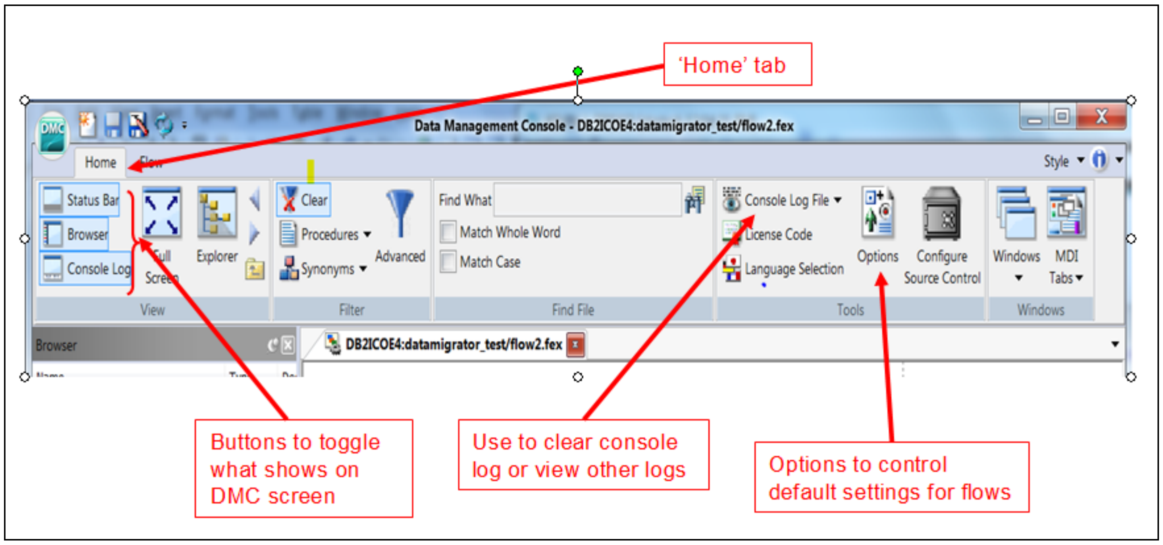


Figure 4: DMC screen showing ribbon interface

DMC has a general layout with the ribbon interface on the top of the window. The ribbon interface provde a set of options that change by context. Depending on what object you have highlighted in DMC, the options in the ribbon can change.

The screen shot shows the ribbon layout when viewing the main *Home* tab. As you start to create flows and other objects, other main tabs will show up, for example *Flow*, and the ribbon will change. At any point while working in the DMC you can click the Home tab and get back to this Home ribbon.

The Home ribbon has a couple of important features that can affect how the DMC works.

➢ The buttons on the side of the ribbon toggle what is shown on the DMC screen.

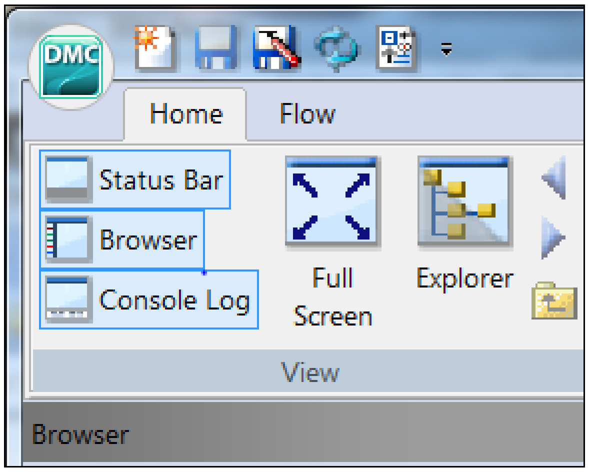


Figure 5: Display Toggle Buttons

For example, if you are working on a DMC flow and the console log is not showing, click on the Console Log button to bring it back. If you notice your DMC window is missing something shown in a screenshot in this guide, use the buttons on the Home ribbon screen to see if you can bring up the 'missing' item.

- As you begin viewing log files, the Console Log File option on the ribbon is useful for clearing the log file or switching to a specific log file.

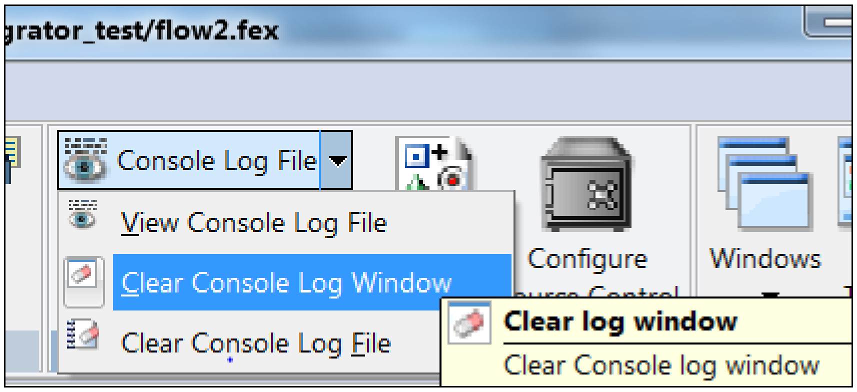
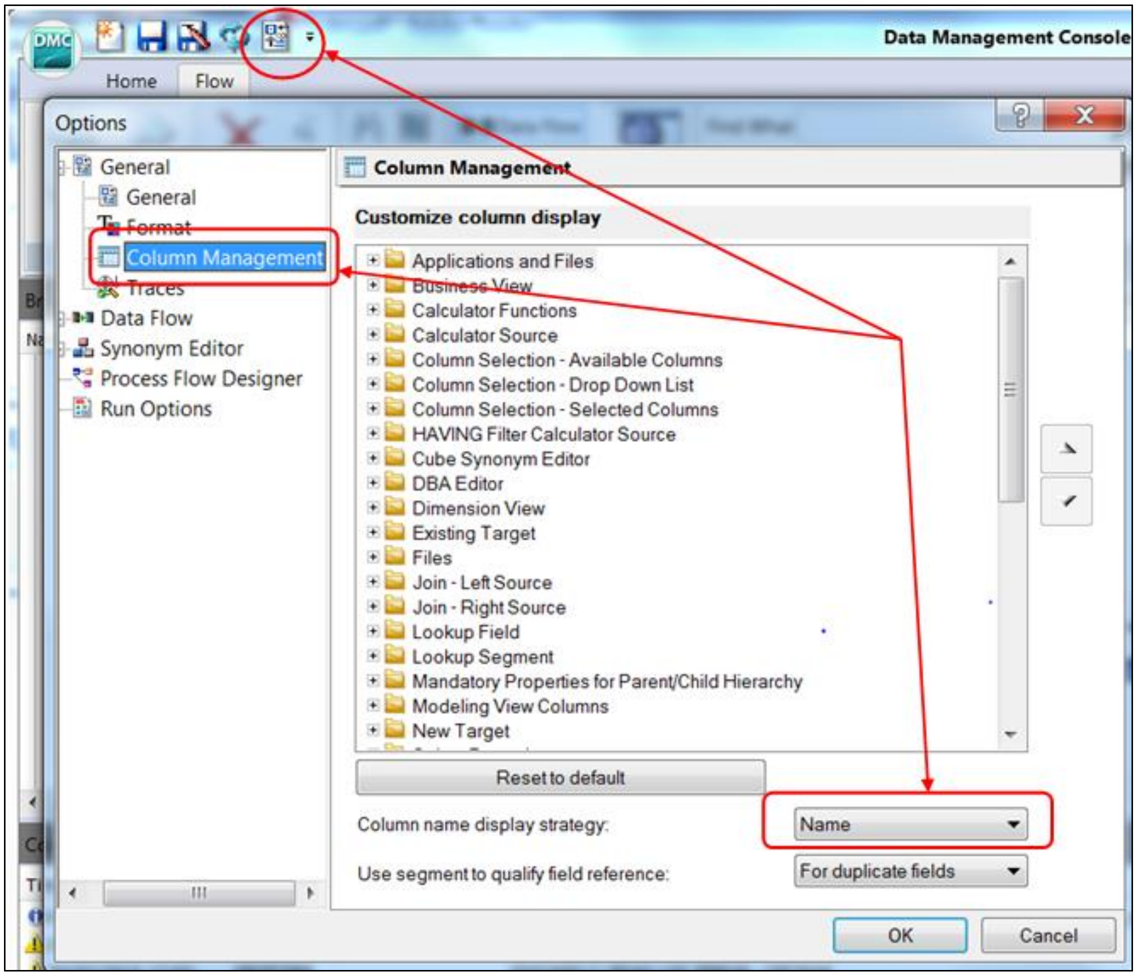


Figure 6: Console Log File control

- Of particular usefulness is the Options button on the ribbon. This brings up a window with a set of options that dictate how DMC works 'by default'. If you find your DMC window looks different than a screenshot, or there is a certain option for which you want to change the default, the Options button is the place to go.

  For example, if you want to change displays to show column names rather than titles when working with tables, choose:
  Options->Column Management->Column name display strategy:



Column name display strategy: Name

Use segment to qualify field reference: For duplicate fields

Figure 7: Changing labels for columns

Another good reference to have is the help text and online guide. They can be found by clicking on the arrow next to the ***i*** information button in the upper right of the DMC window.

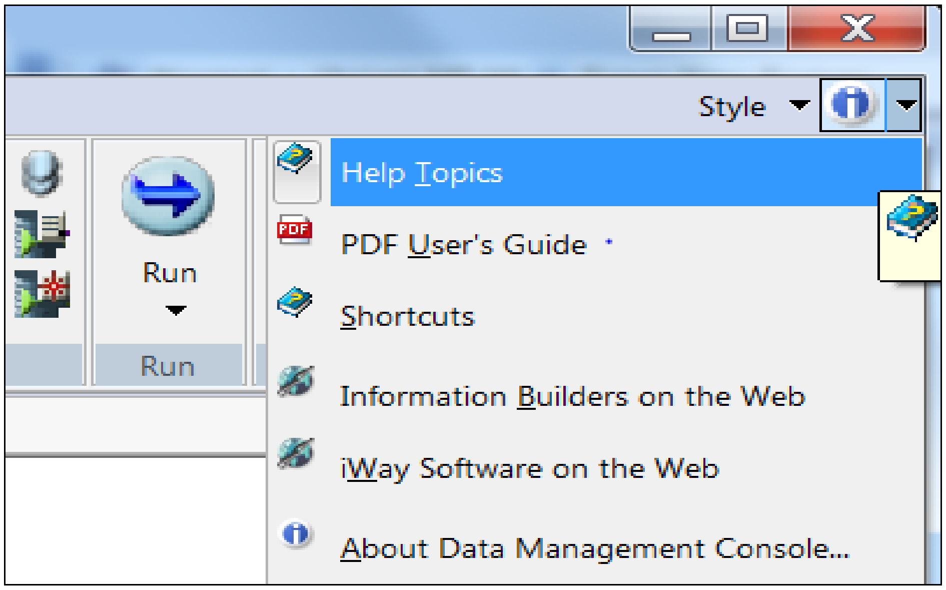


Figure 8: Information drop down menu

Help Topics is a great way to search on a topic for more details. The PDF User's Guide is a full user's guide. It is a more general purpose guide than this user's guide, but has more in depth discussion on certain topics.

With that, let's get back to the steps in creating a flow.

Defining synonyms (Data Sources)

To create a data flow, we first have to have data sources identified. DB2 Web Query for i, which DataMigrator is an extension, comes with a **century** database in library QWQCENT. We will use its files as our data sources.

As a reminder, data sources are also known as synonyms.

DB2 Web Query users know that synonyms can be created through either the BI Portal or Developer Workbench. If we had existing synonyms created through those interfaces we could use those synonyms for DataMigrator as well. However, the DMC also provides a way to create synonyms and we will use that interface to create our synonyms.

Right click on the test folder and specify **New->Synonym**.

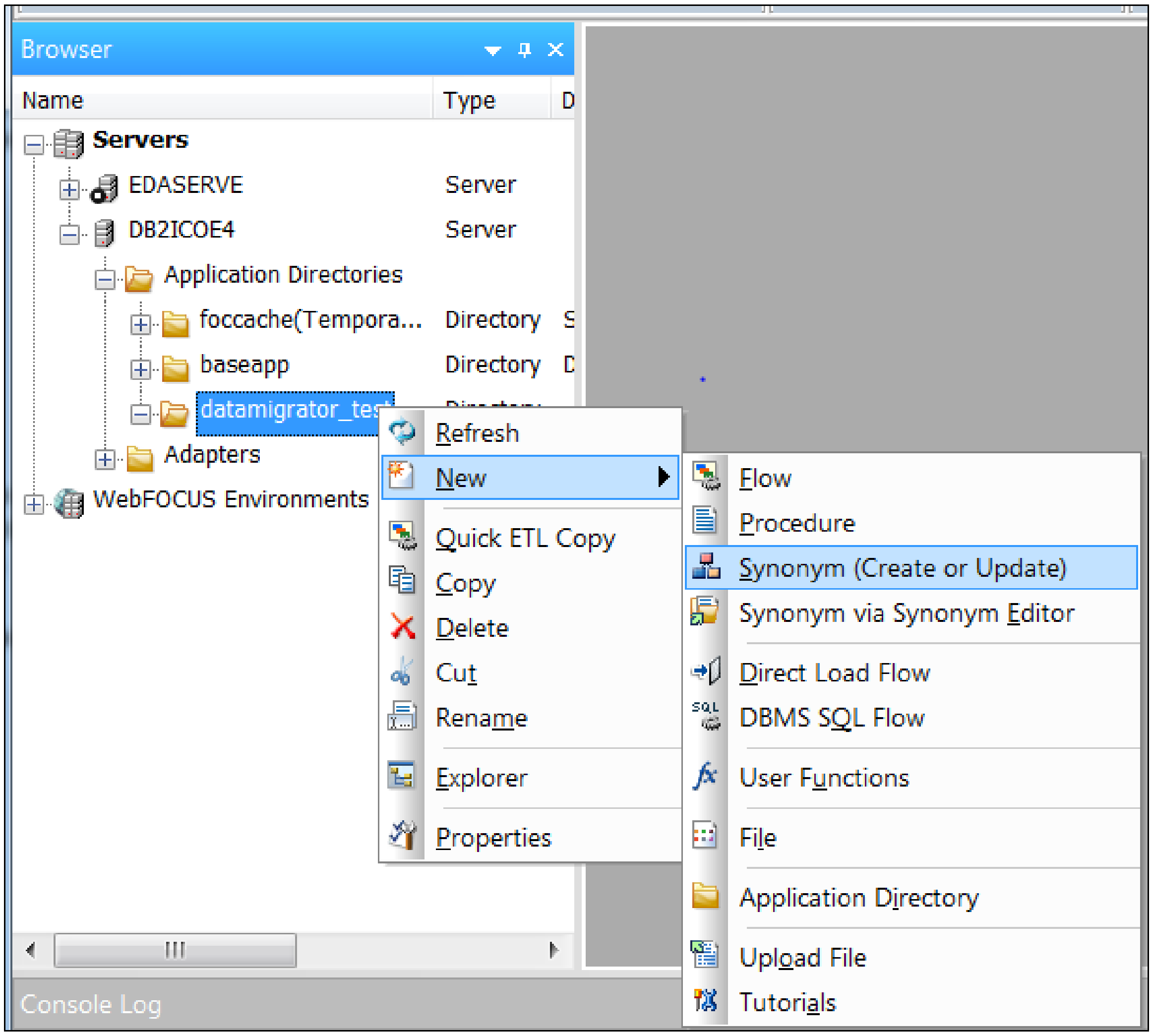


Figure 9: create synonym through DMC

Select the *LOCAL adapter and click **Enter**.

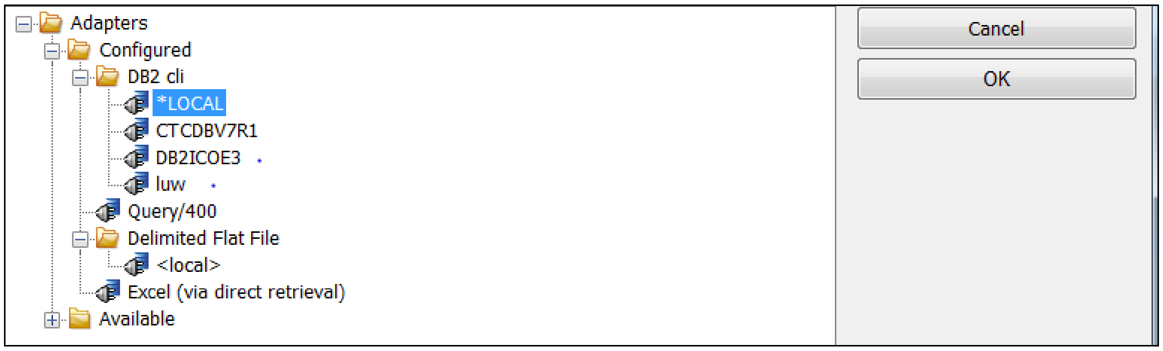
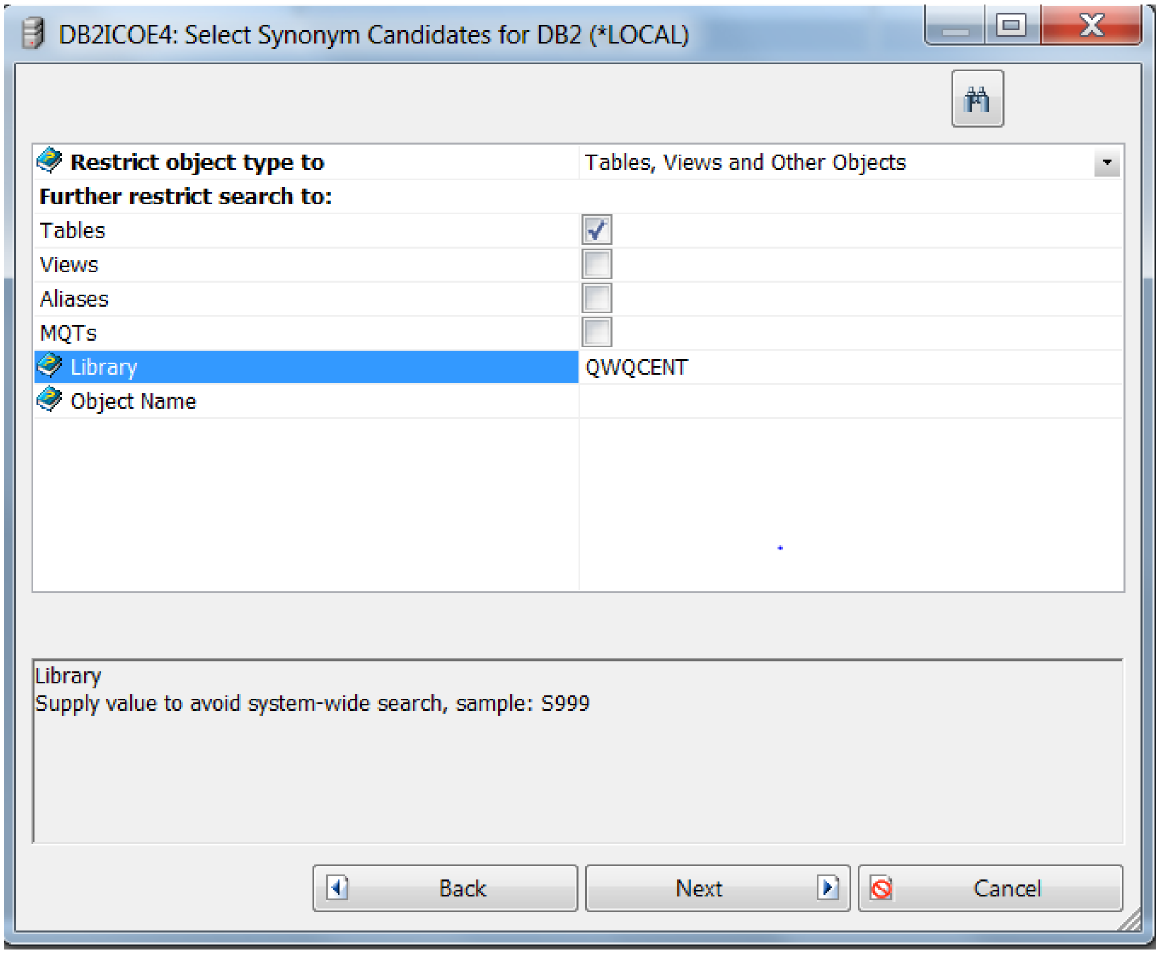


Figure 10: Select *LOCAL adapter

Type in the library **QWQCENT**. Deselect Views, leaving only Tables checked.

Click **Next**.


DB2ICOE4: Select Synonym Candidates for DB2 (*LOCAL)

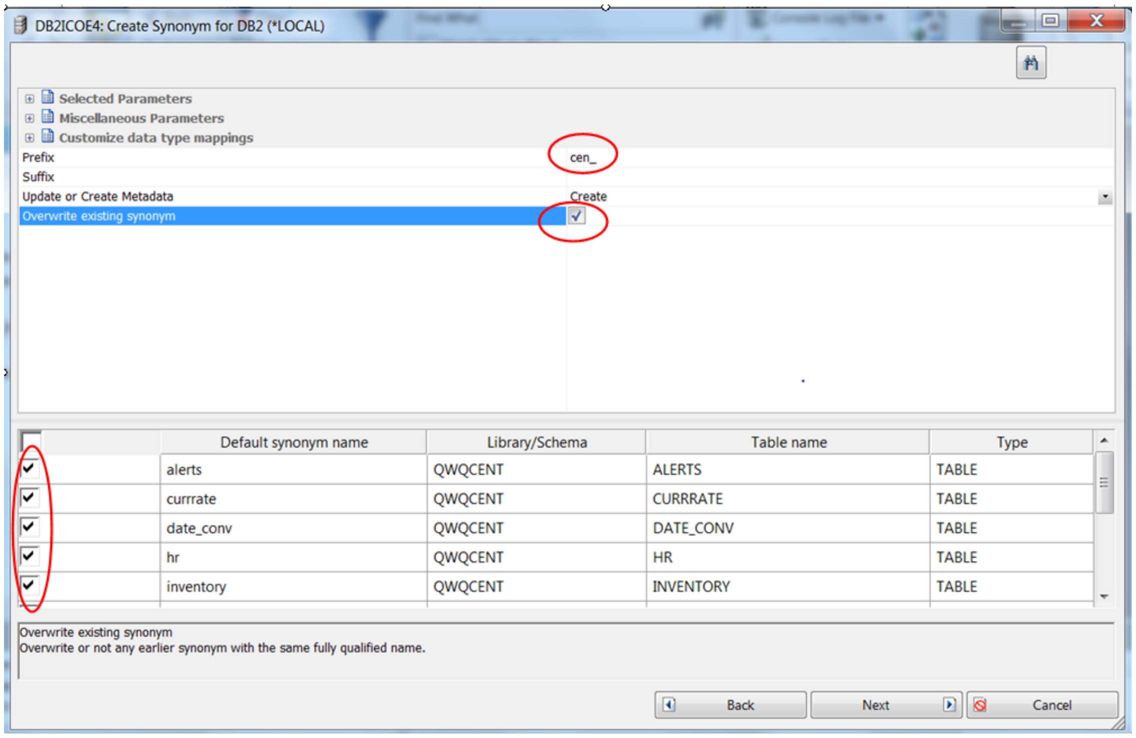
Restrict object type to	Tables, Views and Other Objects
Further restrict search to:	
Tables	✓
Views	
Aliases	
MQTs	
Library	QWQCENT
Object Name	

Library
Supply value to avoid system-wide search, sample: S999

Figure 11: Synonym candidate library QWQCENT - Tables only

Select all the tables shown **except** for *QRPGLESRC*. Following Web Query best practices, provide a prefix to the synonyms, in this case **cen_**. In case synonyms might exist from a previous attempt, specify to *Overwrite existing synonym*.

Click **Next**.


DB2ICOE4: Create Synonym for DB2 (*LOCAL)
Selected Parameters
Miscellaneous Parameters
Customize data type mappings
Back
Next
Cancel

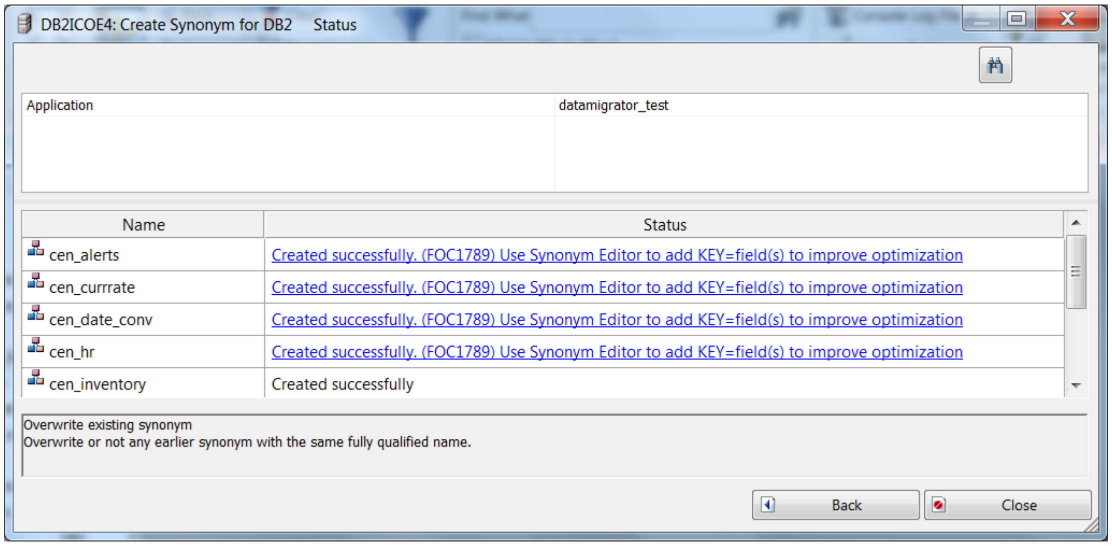
Prefix	cen_
Suffix	
Update or Create Metadata	Create
Overwrite existing synonym	✓

	Default synonym name	Library/Schema	Table name	Type
✓	alerts	QWQCENT	ALERTS	TABLE
✓	currrate	QWQCENT	CURRRATE	TABLE
✓	date_conv	QWQCENT	DATE_CONV	TABLE
✓	hr	QWQCENT	HR	TABLE
✓	inventory	QWQCENT	INVENTORY	TABLE

Overwrite existing synonym
Overwrite or not any earlier synonym with the same fully qualified name.

Figure 12: Create synonyms from tables in QWQCENT.

The synonyms should be created successfully. Close the resulting status window.



Application	datamigrator_test

Name	Status
cen_alerts	Created successfully. (FOC1789) Use Synonym Editor to add KEY=field(s) to improve optimization
cen_currrate	Created successfully. (FOC1789) Use Synonym Editor to add KEY=field(s) to improve optimization
cen_date_conv	Created successfully. (FOC1789) Use Synonym Editor to add KEY=field(s) to improve optimization
cen_hr	Created successfully. (FOC1789) Use Synonym Editor to add KEY=field(s) to improve optimization
cen_inventory	Created successfully

Overwrite existing synonym
Overwrite or not any earlier synonym with the same fully qualified name.

Figure 13: Resulting status window from create synonym.

We are now ready to create a data flow.

Creating a Flow

Right click on the folder and select **New->Flow**

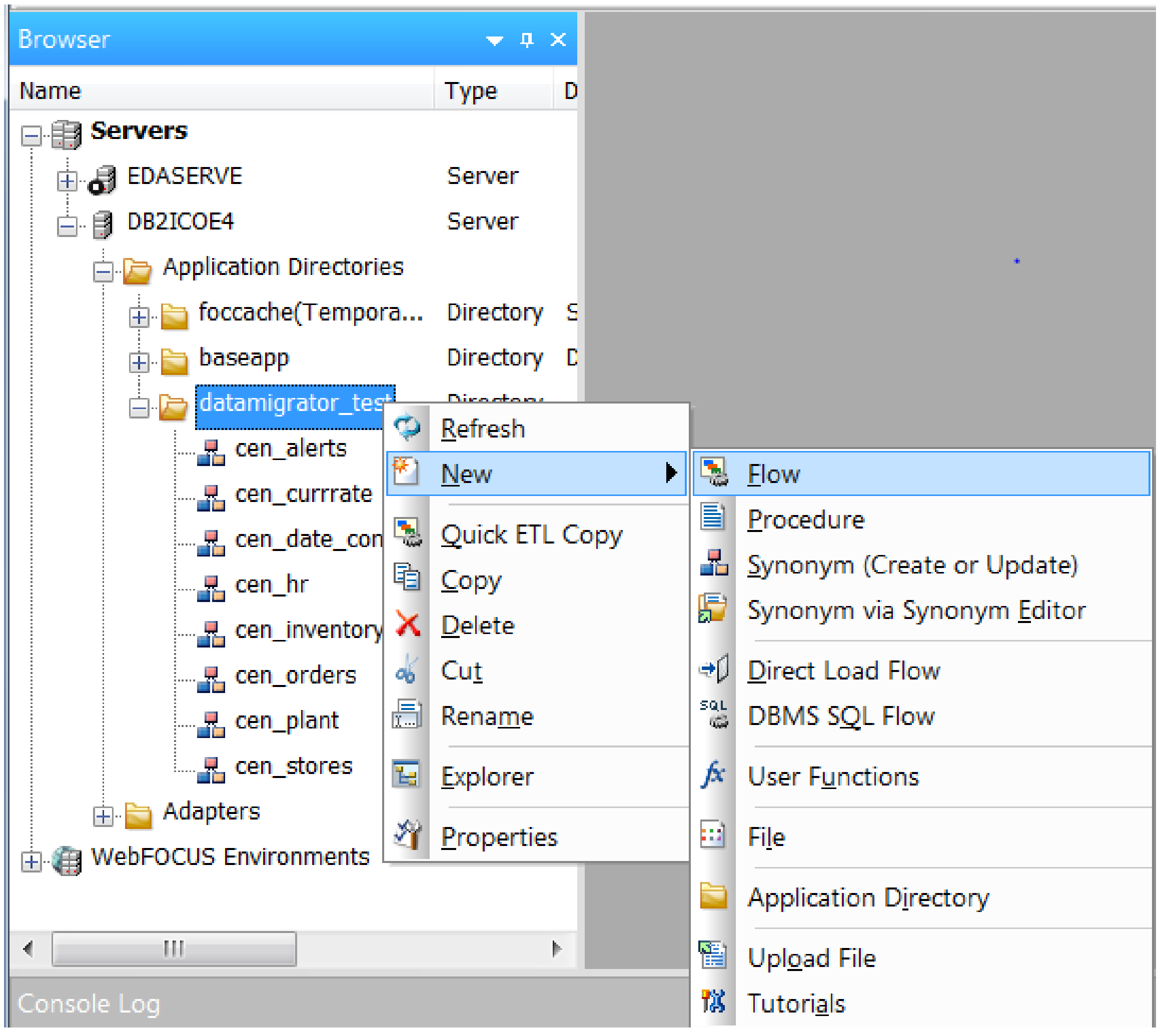


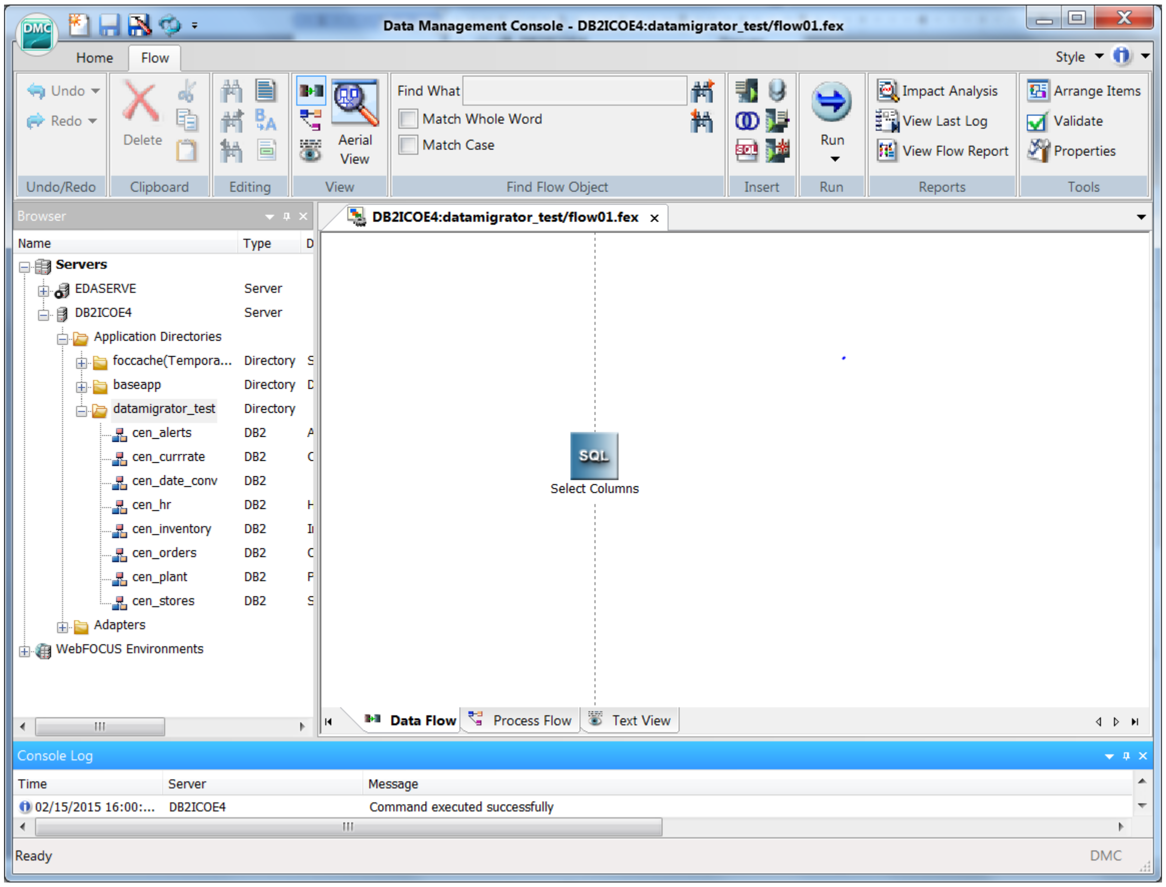
Figure 14: Create new data flow

Note: if you do not see the *Flow* option on the context menu, it means something was not successfully completed during the setup. Either:

1. option 8 of the 5733WQX product is not installed,
2. the enablement Hot Fix was not applied,
3. Web Query was not restarted or
4. the user you signed in as for DMC is not a Developer Workbench user.

Go back to Chapter 2 Setup and Configuration and verify the steps.

The DMC workspace should now show the data flow in a Data Flow tab with the *SQL* icon showing on the palette.



Time	Server	Message
02/15/2015 16:00:...	DB2ICOE4	Command executed successfully

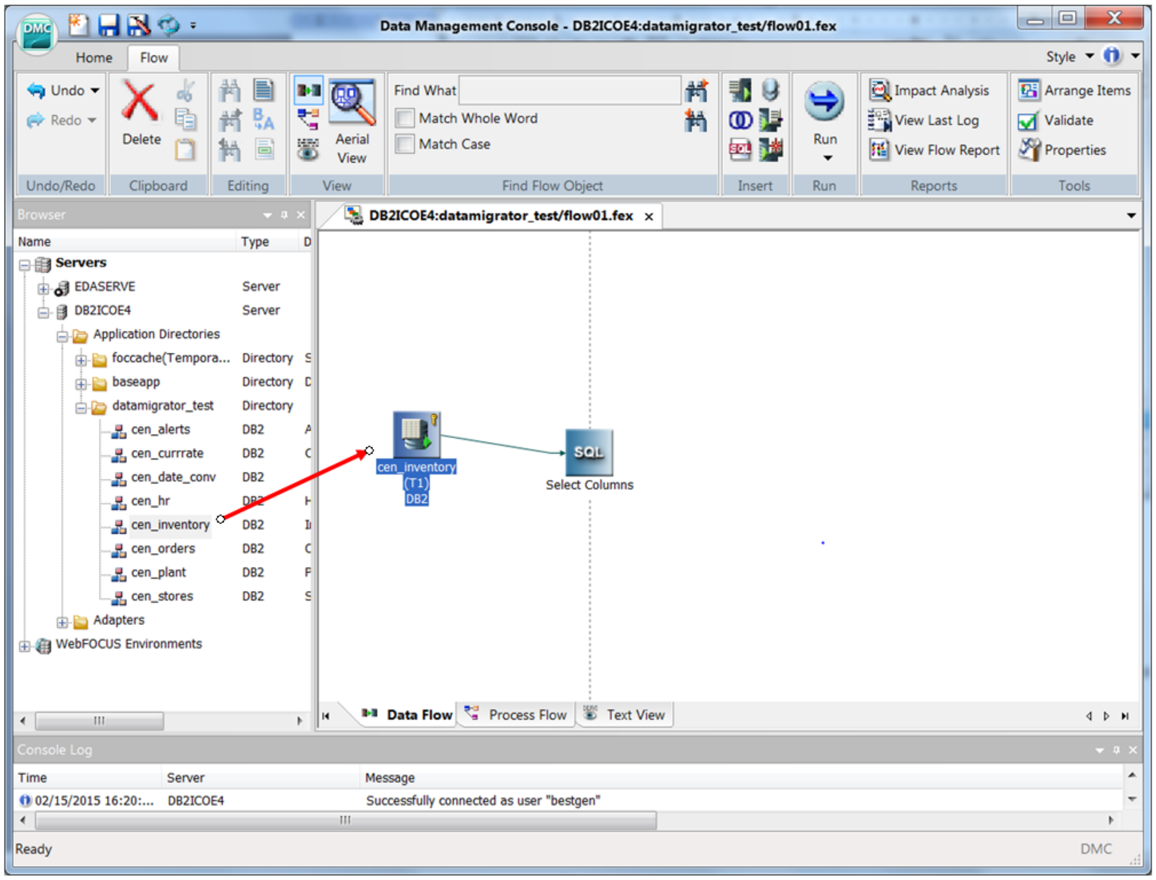
Figure 15: New Data Flow palette

At this point we have the data sources showing in the navigation tree on the left and the data flow on the right.

> **Note:** if the data sources do now show, expand the navigation folders by clicking on the + sign next to the folders.

The *SQL* icon and the dotted vertical line through it represent the major division in the data flow. Objects to the left of the line are data sources. Objects to the right are data targets.

To begin creating the data flow, drag in the data source from the navigation onto the left side of the data flow palette. In this case choose *cen_inventory*.



Name	Type
Servers	
EDASERVE	Server
DB2ICOE4	Server
Application Directories	
foccache(Tempora...	Directory
baseapp	Directory
datamigrator_test	Directory
cen_alerts	DB2
cen_currrate	DB2
cen_date_conv	DB2
cen_hr	DB2
cen_inventory	DB2
cen_orders	DB2
cen_plant	DB2
cen_stores	DB2
Adapters	
WebFOCUS Environments	

```
Time                 Server     Message
02/15/2015 16:20:... DB2ICOE4   Successfully connected as user "bestgen"
```

Figure 16: Choosing data source cen_inventory

When the data source (synonym) is dropped on the palette, DMC automatically connects it to the *SQL* icon with an arrow. This indicates the flow of the data from the data source into the *SQL* operation.

> **Note:** if you happen to drop the synonym to the right side of the SQL icon line it will become a data target, indicated by an arrow pointing into it. It is important to pay attention to which side you drop a synonym on as it could become the target, meaning data would be written into it if you ran the flow!

As an alternative to drag and drop, you can also right click on the left side of the palette or on the SQL icon and choose Add->Source, then Select the synonym for the data source.

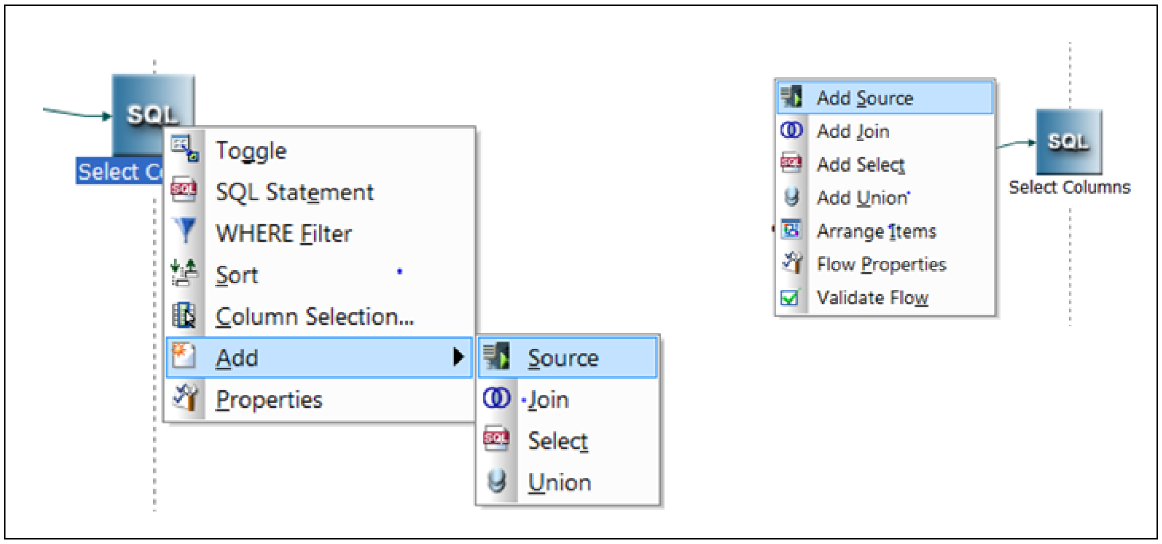


Figure 17: Alternate methods to select a data source

At any point you can sample data in the data source to verify its contents are what you expect. Right click on the data source and choose *Operations->Sample Data*. This will bring up a tab with a data sample from the underlying file. Once you have a general idea of what is in the source file, close the tab with the X as shown.

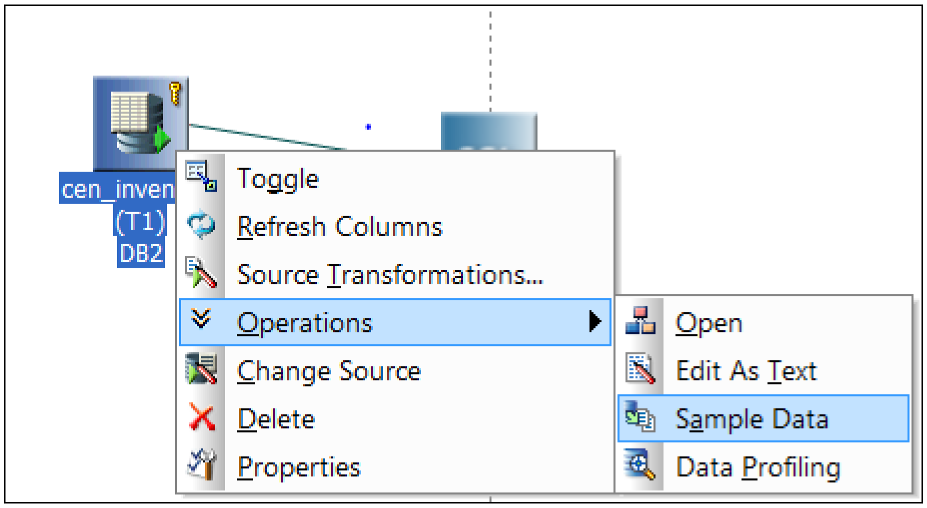
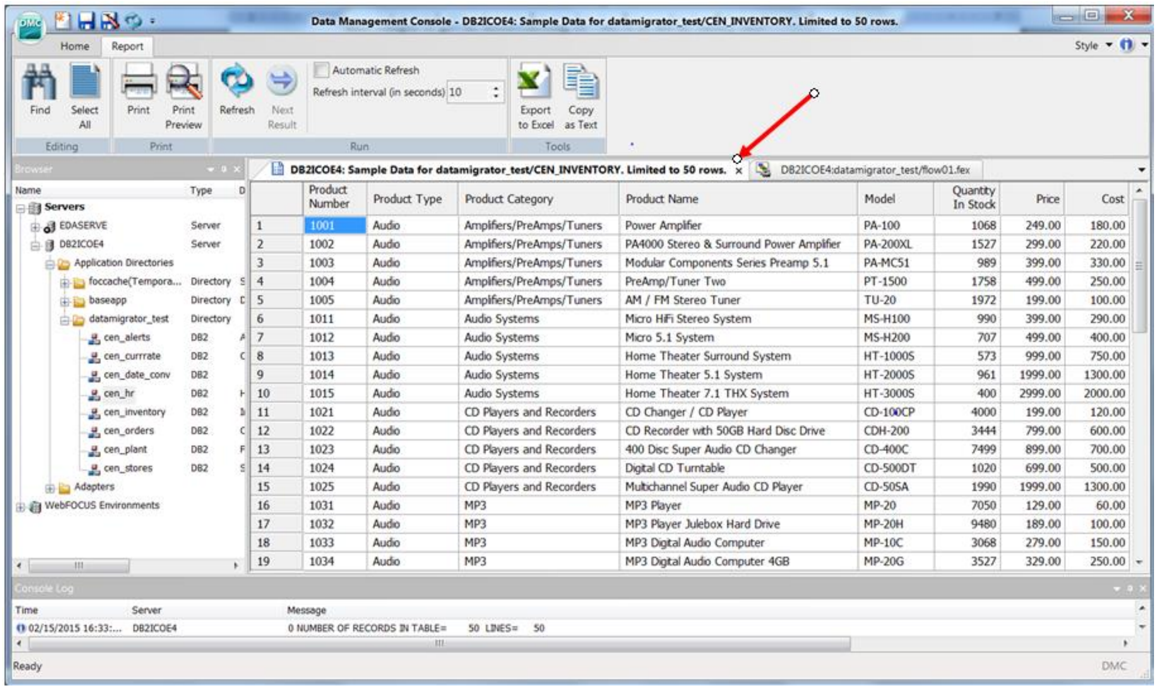


Figure 18: Sample data from a data source



	Product Number	Product Type	Product Category	Product Name	Model	Quantity In Stock	Price	Cost
1	1001	Audio	Amplifiers/PreAmps/Tuners	Power Amplifier	PA-100	1068	249.00	180.00
2	1002	Audio	Amplifiers/PreAmps/Tuners	PA4000 Stereo & Surround Power Amplifier	PA-200XL	1527	299.00	220.00
3	1003	Audio	Amplifiers/PreAmps/Tuners	Modular Components Series Preamp 5.1	PA-MC51	989	399.00	330.00
4	1004	Audio	Amplifiers/PreAmps/Tuners	PreAmp/Tuner Two	PT-1500	1758	499.00	250.00
5	1005	Audio	Amplifiers/PreAmps/Tuners	AM / FM Stereo Tuner	TU-20	1972	199.00	100.00
6	1011	Audio	Audio Systems	Micro HiFi Stereo System	MS-H100	990	399.00	290.00
7	1012	Audio	Audio Systems	Micro 5.1 System	MS-H200	707	499.00	400.00
8	1013	Audio	Audio Systems	Home Theater Surround System	HT-1000S	573	999.00	750.00
9	1014	Audio	Audio Systems	Home Theater 5.1 System	HT-2000S	961	1999.00	1300.00
10	1015	Audio	Audio Systems	Home Theater 7.1 THX System	HT-3000S	400	2999.00	2000.00
11	1021	Audio	CD Players and Recorders	CD Changer / CD Player	CD-100CP	4000	199.00	120.00
12	1022	Audio	CD Players and Recorders	CD Recorder with 50GB Hard Disc Drive	CDH-200	3444	799.00	600.00
13	1023	Audio	CD Players and Recorders	400 Disc Super Audio CD Changer	CD-400C	7499	899.00	700.00
14	1024	Audio	CD Players and Recorders	Digital CD Turntable	CD-500DT	1020	699.00	500.00
15	1025	Audio	CD Players and Recorders	Multichannel Super Audio CD Player	CD-50SA	1990	1999.00	1300.00
16	1031	Audio	MP3	MP3 Player	MP-20	7050	129.00	60.00
17	1032	Audio	MP3	MP3 Player Julebox Hard Drive	MP-20H	9480	189.00	100.00
18	1033	Audio	MP3	MP3 Digital Audio Computer	MP-10C	3068	279.00	150.00
19	1034	Audio	MP3	MP3 Digital Audio Computer 4GB	MP-20G	3527	329.00	250.00

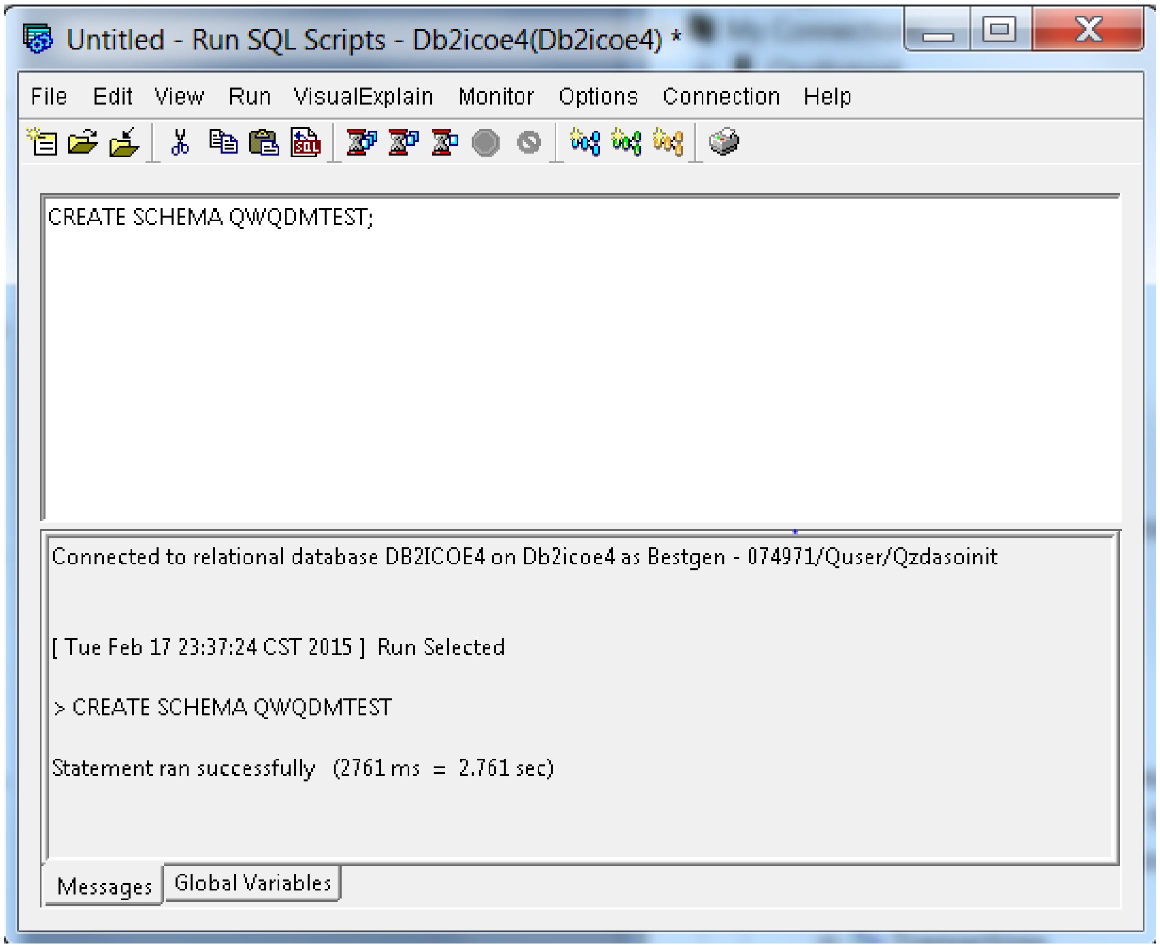
Time	Server	Message
02/15/2015 16:33:...	DB2ICOE4	0 NUMBER OF RECORDS IN TABLE= 50 LINES= 50

Figure 19: Sample data tab, highlighting the x that will close the sample data tab

For now, let’s complete a simple flow by adding a data target.

Adding Data Target

We will create a library to contain the resulting database table. Using System i Navigator, create an SQL schema called **QWQDMTEST**. Alternatively, using a CL command line, you may use STRSQL to create the schema.



```
CREATE SCHEMA QWQDMTEST;
```

```
Connected to relational database DB2ICOE4 on Db2icoe4 as Bestgen - 074971/Quser/Qzdasoinit

[ Tue Feb 17 23:37:24 CST 2015 ]  Run Selected

> CREATE SCHEMA QWQDMTEST

Statement ran successfully   (2761 ms  =  2.761 sec)
```

Figure 20: Create schema QWQDMTEST

Now switch back to DMC.

> **Note:** DataMigrator has the ability to create a new data target table or insert into an existing target. Depending on your intended use, you may want anything from a simple copy of the source, frequently called an ODS, to something more involved where the target is a full designed database, such as a Data Warehouse or Data Mart.

Add a data target. Right click on the right side of the palette and choose *Add Target->New*. Alternatively, you can click on *New Target* in the ribbon and drag it onto the workspace.

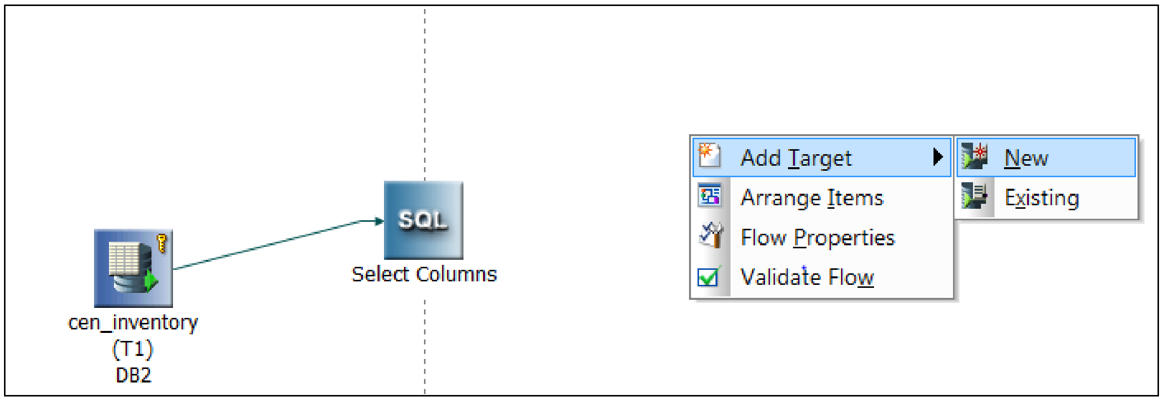


Figure 21: Right click on palette to add new target

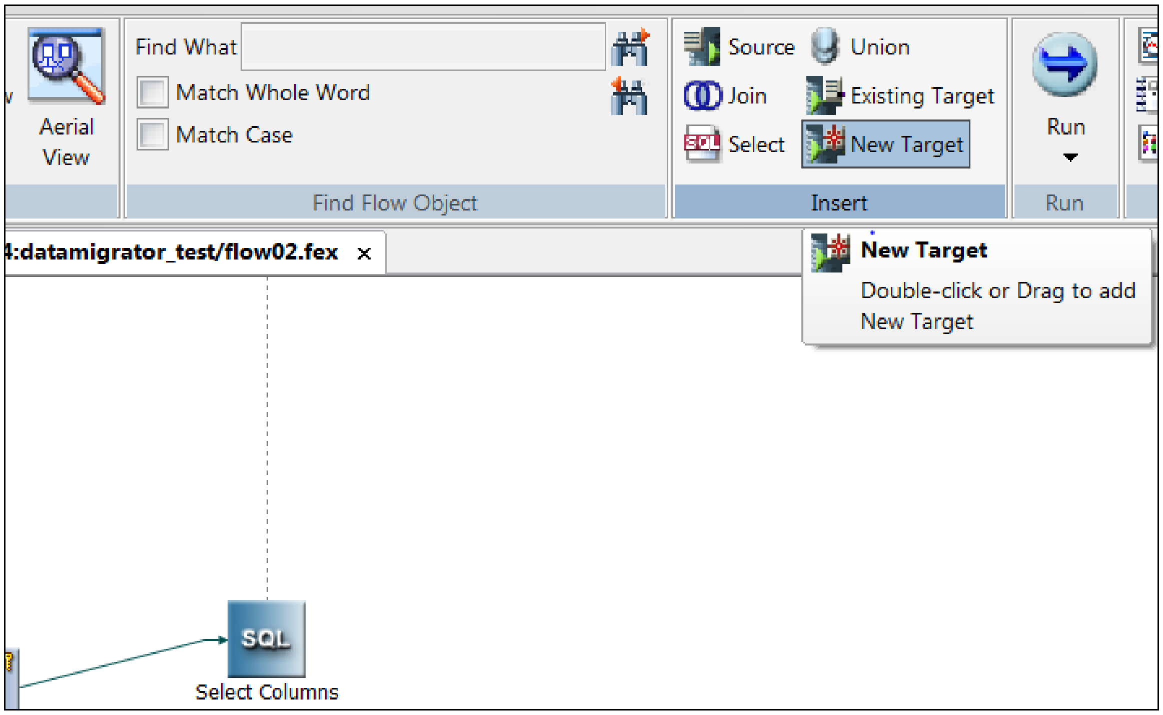


Figure 22: Alternative - add new target from ribbon

Once the data target is placed on the workspace, right click on the target icon and select Properties.

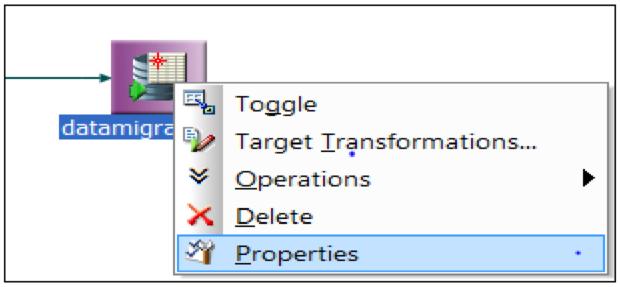
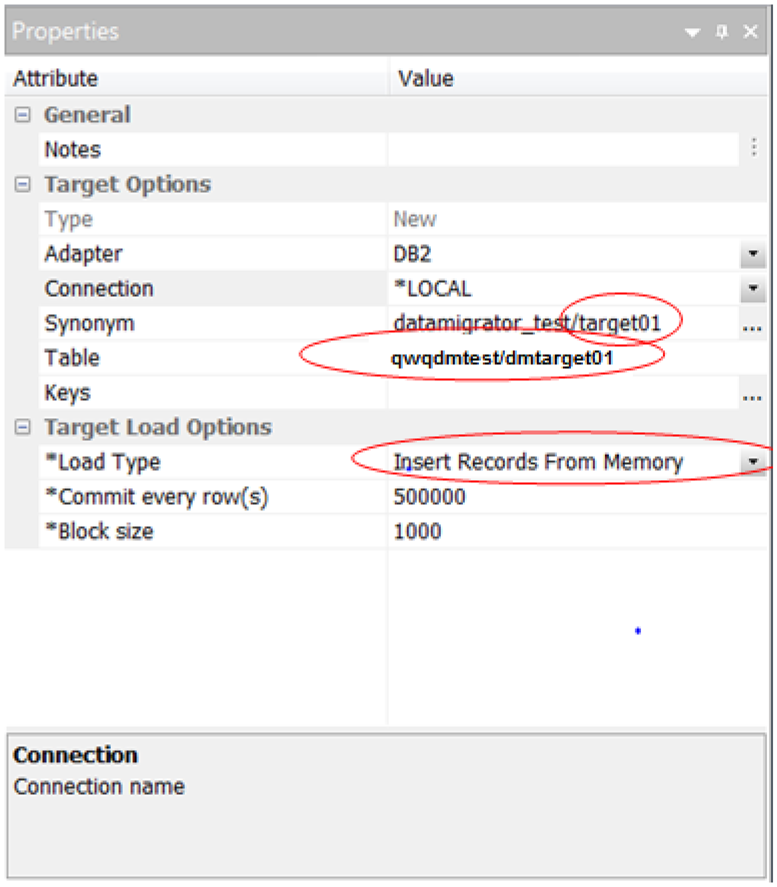


Figure 23: Select Properties from data target

The **Properties** dialog provides a way to control aspects of the underlying database target table and how it is populated. There are three main items of interest at this point:

1. *Synonym* name - By default a synonym will be created (during the run), named targetxx (where xx is an increasing number) and will reside in the folder where the flow was created. This can be change by choosing the ellipsis to the right of the text. We will leave it as is.
2. *Table* – The underlying database table name and location. By default the database table will be the same name as the synonym, targetxx in this case. It will also not be library qualified, which means the database table will be created in the first writeable library in your user's library list. The library can be explicitly specified using the lib/file IBM i notation. For this example, change the table library and name to **qwqdmtest/dmtarget01**.
3. *Load Type* - This determines how the records are read from the data source(s) and written to the table. By default this is Insert/Update. However, for the 'bulk load' which we will be doing, choosing *Insert Records From Memory* is a better performing choice as it allows more record blocking. Choose that option.



Attribute	Value
General	
Notes	
Target Options	
Type	New
Adapter	DB2
Connection	*LOCAL
Synonym	datamigrator_test/target01
Table	qwqdmtest/dmtarget01
Keys	
Target Load Options	
*Load Type	Insert Records From Memory
*Commit every row(s)	500000
*Block size	1000

Connection
Connection name

Figure 24: setting properties for data target qwqdmtest/dmtarget01

Note: The Properties dialog changes depending on what item or icon you have highlighted. If you find that the Properties do not look like above, (re)click on the data target icon to bring this Properties context back into focus.

Running a Flow

At this point, let’s attempt to run this simple data flow. Click on the Run option in the ribbon above the workspace.

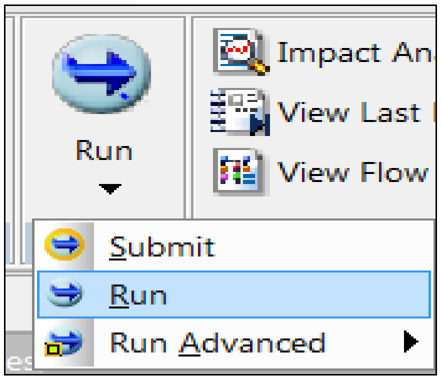
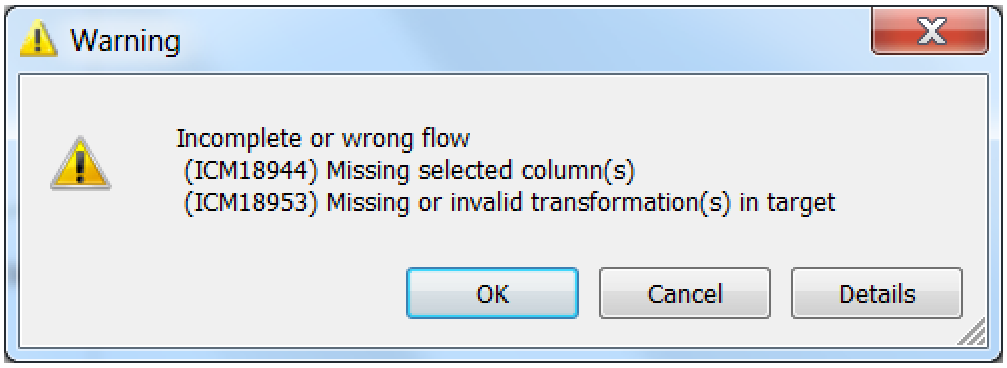


Figure 25: Run flow option

An error is given stating columns are missing.


Warning

Incomplete or wrong flow
(ICM18944) Missing selected column(s)
(ICM18953) Missing or invalid transformation(s) in target

Figure 26: Missing selected columns

This error highlights an important distinction in DataMigrator for i compared to a simple file copy tool. Because it is an ETL product and not just a file copy tool, DataMigrator assumes you will be doing some amount of transformation or selection in the data flow. In this case, we defined a source and a target, but no column mapping or data transformation was given. Even though DataMigrator **can** be used for simple file copies, it really is built to do more than that[1].

The *SQL* icon signifies the transition from the source to the target. It is a major point where mapping and transformation can be defined. We will utilize it now.

[1] That said, the default behavior for including columns can be overridden via ‘Options’

Let's go back and add the list of columns we want mapped from the data source into the target. Right click on the *SQL* icon and choose *Column Selection*.

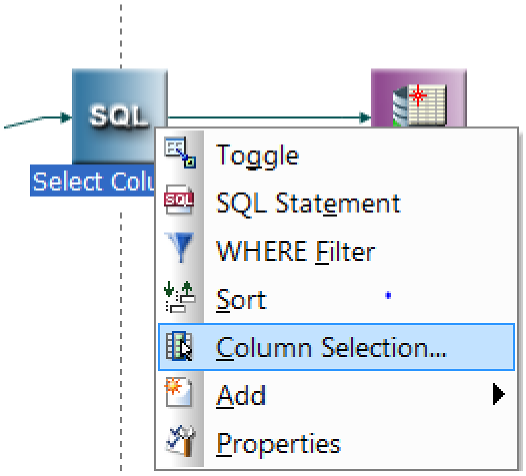
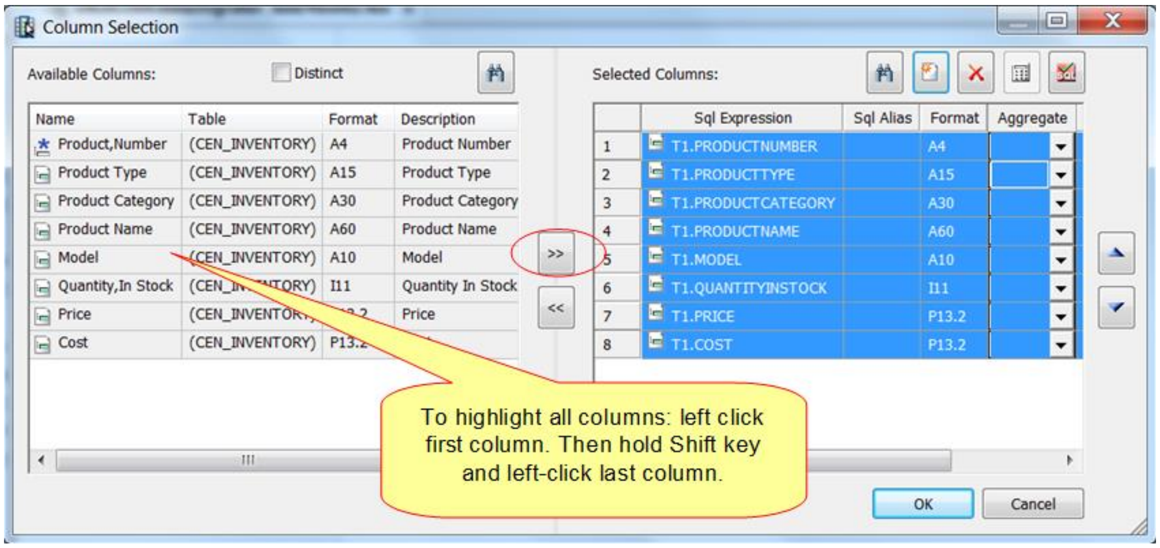


Figure 27: Column selection option

The *Column Selection* window appears. The left side shows all the potential columns available from the data source. Highlight all columns and 'Select' them by clicking the >> button in the middle of the window. Alternatively, select each column one at a time and select it with the >> button.


Column Selection
Available Columns:
Distinct
Selected Columns:
>>
<<
To highlight all columns: left click first column. Then hold Shift key and left-click last column.
OK
Cancel

Name	Table	Format	Description
Product,Number	(CEN_INVENTORY)	A4	Product Number
Product Type	(CEN_INVENTORY)	A15	Product Type
Product Category	(CEN_INVENTORY)	A30	Product Category
Product Name	(CEN_INVENTORY)	A60	Product Name
Model	(CEN_INVENTORY)	A10	Model
Quantity,In Stock	(CEN_INVENTORY)	I11	Quantity In Stock
Price	(CEN_INVENTORY)	[illegible]	Price
Cost	(CEN_INVENTORY)	P13.2	[illegible]

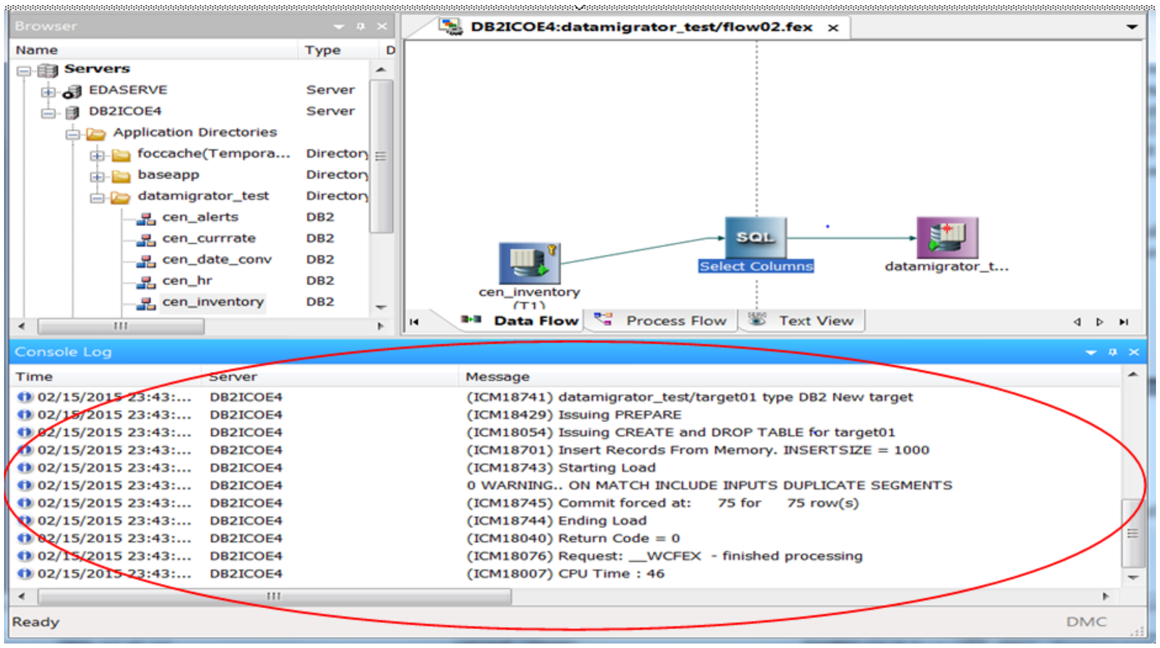
	Sql Expression	Sql Alias	Format	Aggregate
1	T1.PRODUCTNUMBER		A4	
2	T1.PRODUCTTYPE		A15	
3	T1.PRODUCTCATEGORY		A30	
4	T1.PRODUCTNAME		A60	
5	T1.MODEL		A10	
6	T1.QUANTITYINSTOCK		I11	
7	T1.PRICE		P13.2	
8	T1.COST		P13.2	

Figure 28: selecting columns from the data source

Before continuing, examine the Column Selection window. Note that there are several powerful capabilities available with a simple click. For example, duplicate rows can be eliminated during the run of the flow by checking the *Distinct* option at the top. Also, the order in which the columns appear in the target can be rearranged by simply highlighting a selected column and using the up/down arrows to move it around. More complex operations like expression and aggregate (group by) processing can also be specified. We will revisit these in more detail in a later section.

Once all columns have been selected, click OK on the Column Selection window. Click again on the Run option in the ribbon above the workspace. This time the flows should run successfully.

Look at the Console log at the bottom of the DMC screen to verify it worked.

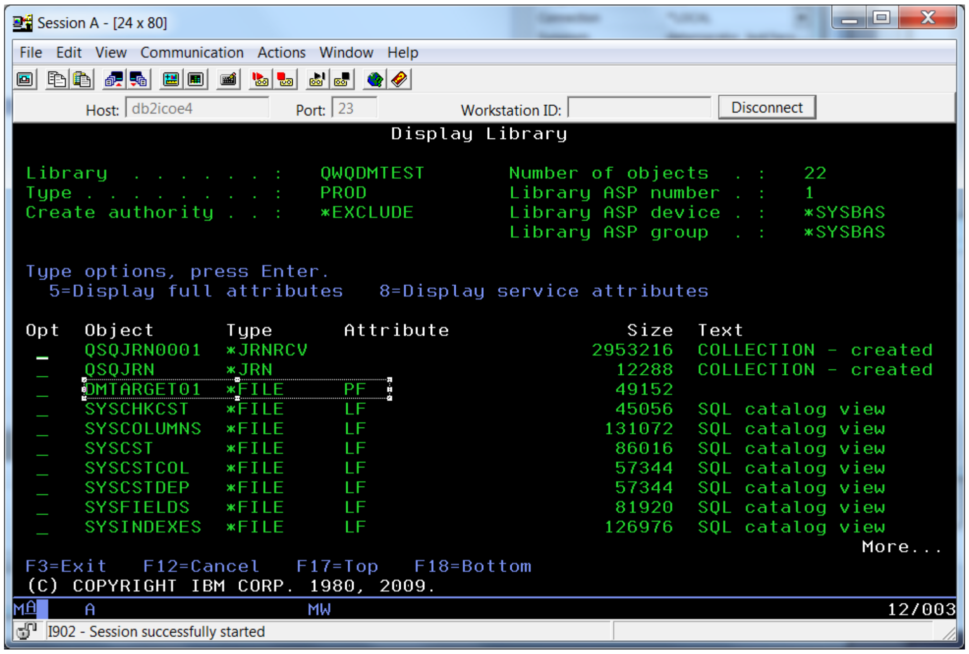


```
Time                   Server      Message
02/15/2015 23:43:...   DB2ICOE4    (ICM18741) datamigrator_test/target01 type DB2 New target
02/15/2015 23:43:...   DB2ICOE4    (ICM18429) Issuing PREPARE
02/15/2015 23:43:...   DB2ICOE4    (ICM18054) Issuing CREATE and DROP TABLE for target01
02/15/2015 23:43:...   DB2ICOE4    (ICM18701) Insert Records From Memory. INSERTSIZE = 1000
02/15/2015 23:43:...   DB2ICOE4    (ICM18743) Starting Load
02/15/2015 23:43:...   DB2ICOE4    0 WARNING.. ON MATCH INCLUDE INPUTS DUPLICATE SEGMENTS
02/15/2015 23:43:...   DB2ICOE4    (ICM18745) Commit forced at:     75 for     75 row(s)
02/15/2015 23:43:...   DB2ICOE4    (ICM18744) Ending Load
02/15/2015 23:43:...   DB2ICOE4    (ICM18040) Return Code = 0
02/15/2015 23:43:...   DB2ICOE4    (ICM18076) Request: __WCFEX  - finished processing
02/15/2015 23:43:...   DB2ICOE4    (ICM18007) CPU Time : 46
```

Figure 29: Console log showing successful run of flow

Verify the target table was created and populated using three methods:

1. Look in the qwqdmtest library to verify the file was created.
2. Look at the target file contents from outside the DMC e.g. SQL.
3. Use the Sample Data option from the data target icon (right click).


Session A - [24 x 80]

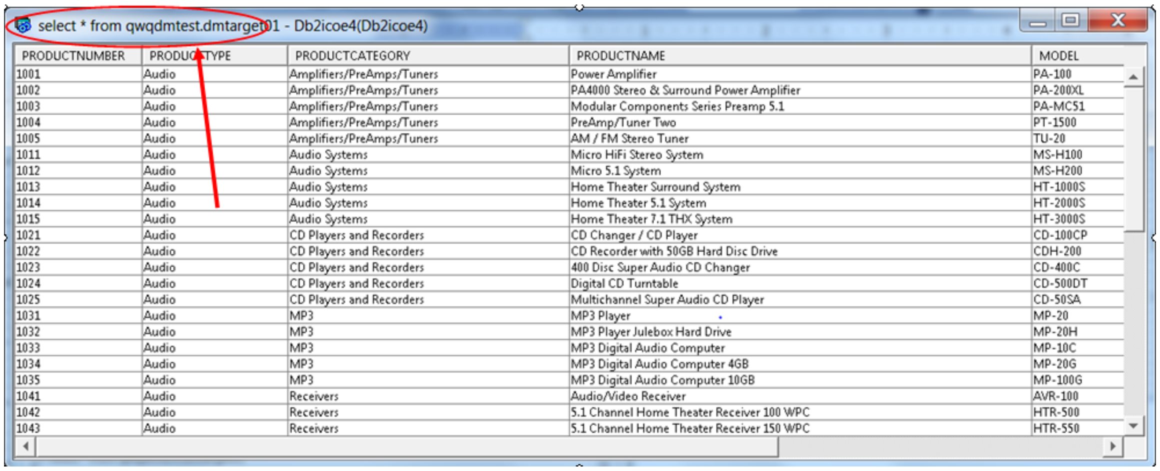
File Edit View Communication Actions Window Help

Host: db2icoe4 Port: 23 Workstation ID: Disconnect

```
                                Display Library

 Library  . . . . . . :   QWQDMTEST        Number of objects  . :   22
 Type . . . . . . . . :   PROD             Library ASP number . :   1
 Create authority . . :   *EXCLUDE         Library ASP device . :   *SYSBAS
                                           Library ASP group  . :   *SYSBAS

 Type options, press Enter.
   5=Display full attributes   8=Display service attributes

 Opt  Object      Type       Attribute             Size  Text
  _   QSQJRN0001  *JRNRCV                       2953216  COLLECTION - created
  _   QSQJRN      *JRN                            12288  COLLECTION - created
  _   DMTARGET01  *FILE      PF                   49152
  _   SYSCHKCST   *FILE      LF                   45056  SQL catalog view
  _   SYSCOLUMNS  *FILE      LF                  131072  SQL catalog view
  _   SYSCST      *FILE      LF                   86016  SQL catalog view
  _   SYSCSTCOL   *FILE      LF                   57344  SQL catalog view
  _   SYSCSTDEP   *FILE      LF                   57344  SQL catalog view
  _   SYSFIELDS   *FILE      LF                   81920  SQL catalog view
  _   SYSINDEXES  *FILE      LF                  126976  SQL catalog view
                                                                        More...
 F3=Exit   F12=Cancel   F17=Top   F18=Bottom
 (C) COPYRIGHT IBM CORP. 1980, 2009.
MA     A                  MW                                            12/003
```

I902 - Session successfully started

Figure 30: Data target table created


select * from qwqdmtest.dmtarget01 - Db2icoe4(Db2icoe4)

PRODUCTNUMBER	PRODUCTTYPE	PRODUCTCATEGORY	PRODUCTNAME	MODEL
1001	Audio	Amplifiers/PreAmps/Tuners	Power Amplifier	PA-100
1002	Audio	Amplifiers/PreAmps/Tuners	PA4000 Stereo & Surround Power Amplifier	PA-200XL
1003	Audio	Amplifiers/PreAmps/Tuners	Modular Components Series Preamp 5.1	PA-MC51
1004	Audio	Amplifiers/PreAmps/Tuners	PreAmp/Tuner Two	PT-1500
1005	Audio	Amplifiers/PreAmps/Tuners	AM / FM Stereo Tuner	TU-20
1011	Audio	Audio Systems	Micro HiFi Stereo System	MS-H100
1012	Audio	Audio Systems	Micro 5.1 System	MS-H200
1013	Audio	Audio Systems	Home Theater Surround System	HT-1000S
1014	Audio	Audio Systems	Home Theater 5.1 System	HT-2000S
1015	Audio	Audio Systems	Home Theater 7.1 THX System	HT-3000S
1021	Audio	CD Players and Recorders	CD Changer / CD Player	CD-100CP
1022	Audio	CD Players and Recorders	CD Recorder with 50GB Hard Disc Drive	CDH-200
1023	Audio	CD Players and Recorders	400 Disc Super Audio CD Changer	CD-400C
1024	Audio	CD Players and Recorders	Digital CD Turntable	CD-500DT
1025	Audio	CD Players and Recorders	Multichannel Super Audio CD Player	CD-50SA
1031	Audio	MP3	MP3 Player	MP-20
1032	Audio	MP3	MP3 Player Julebox Hard Drive	MP-20H
1033	Audio	MP3	MP3 Digital Audio Computer	MP-10C
1034	Audio	MP3	MP3 Digital Audio Computer 4GB	MP-20G
1035	Audio	MP3	MP3 Digital Audio Computer 10GB	MP-100G
1041	Audio	Receivers	Audio/Video Receiver	AVR-100
1042	Audio	Receivers	5.1 Channel Home Theater Receiver 100 WPC	HTR-500
1043	Audio	Receivers	5.1 Channel Home Theater Receiver 150 WPC	HTR-550

Figure 31: Verifying contents of data target table via SQL

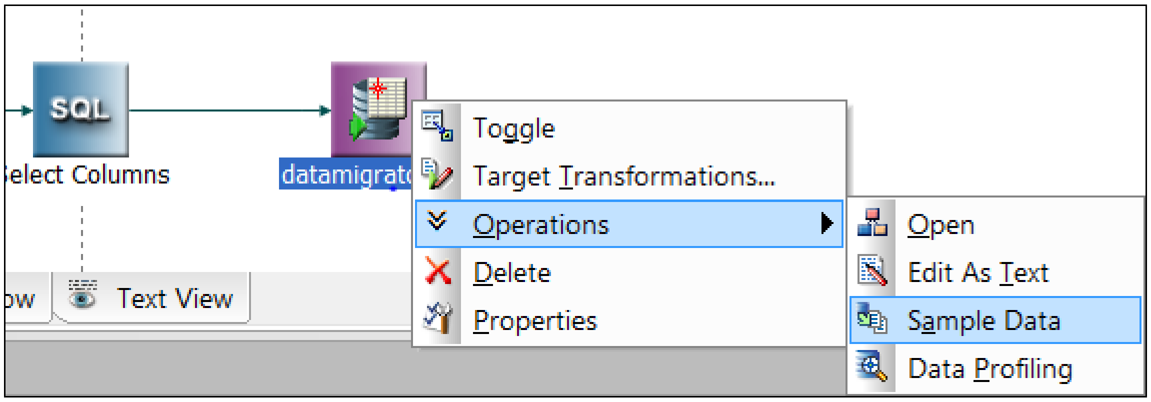


Figure 32: Sample data from the data target

Saving a Flow

Save this data flow. Call it *flow1* and save it into the datamigrator_test folder.

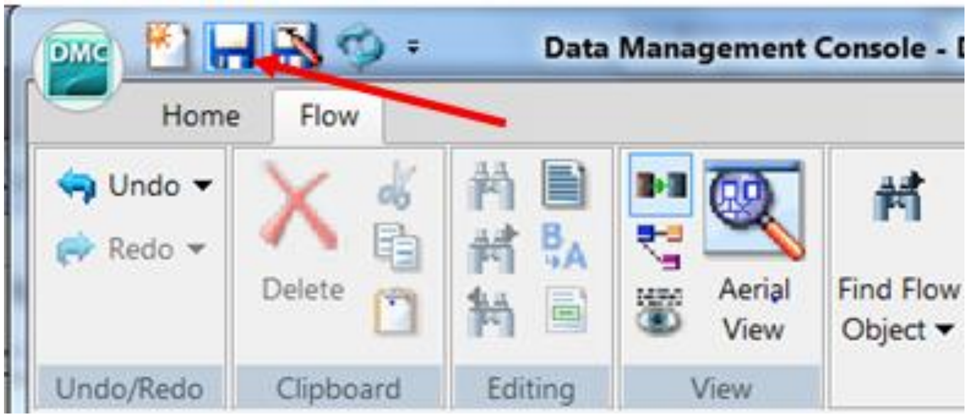


Figure 33: Flow save option

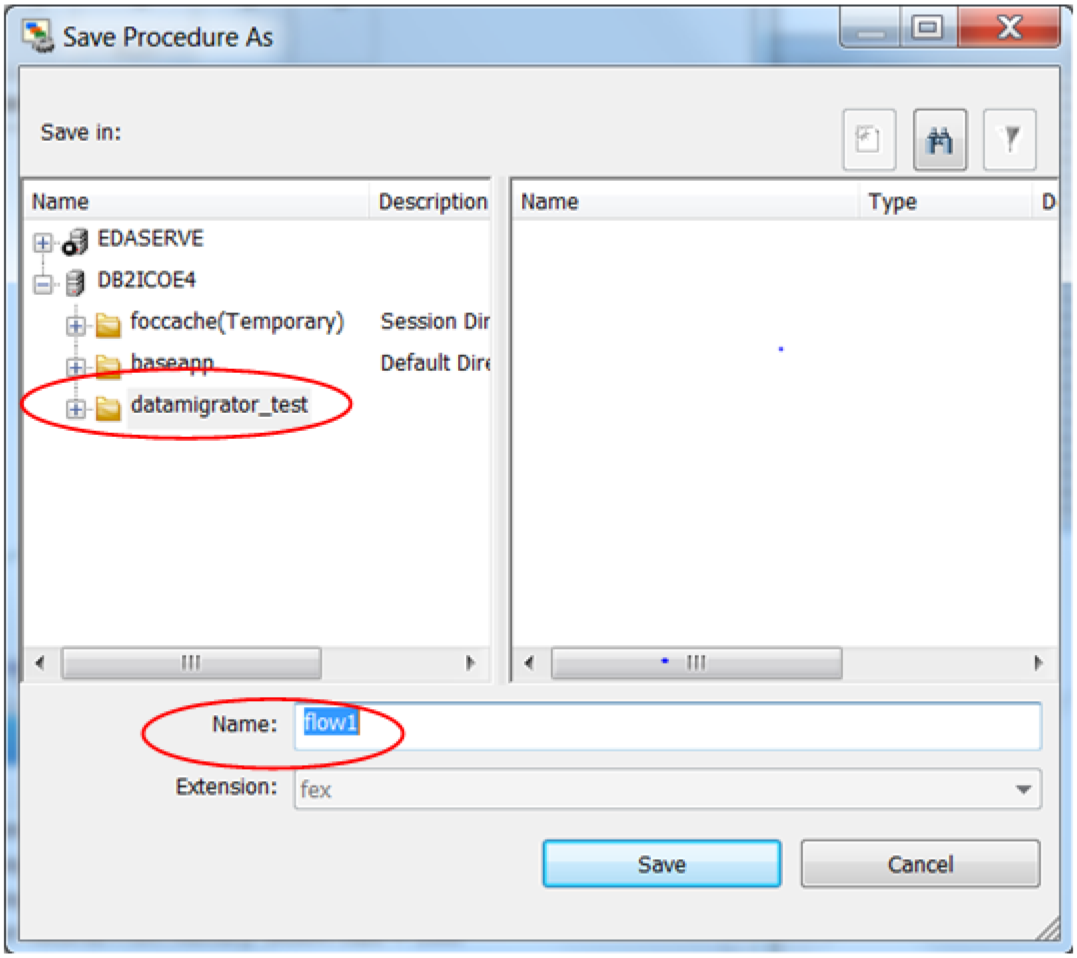


Figure 34: Saving flow1

Close out the flow1 tab to clean up the workspace.

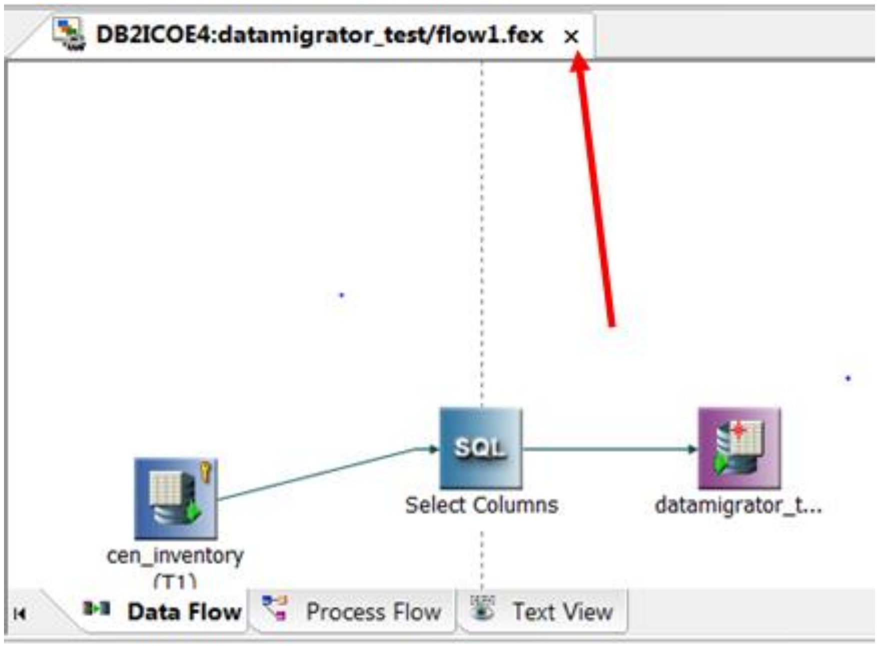


Figure 35: Closing out dataflow tab

Chapter 4. A Deeper Look at Data Flows

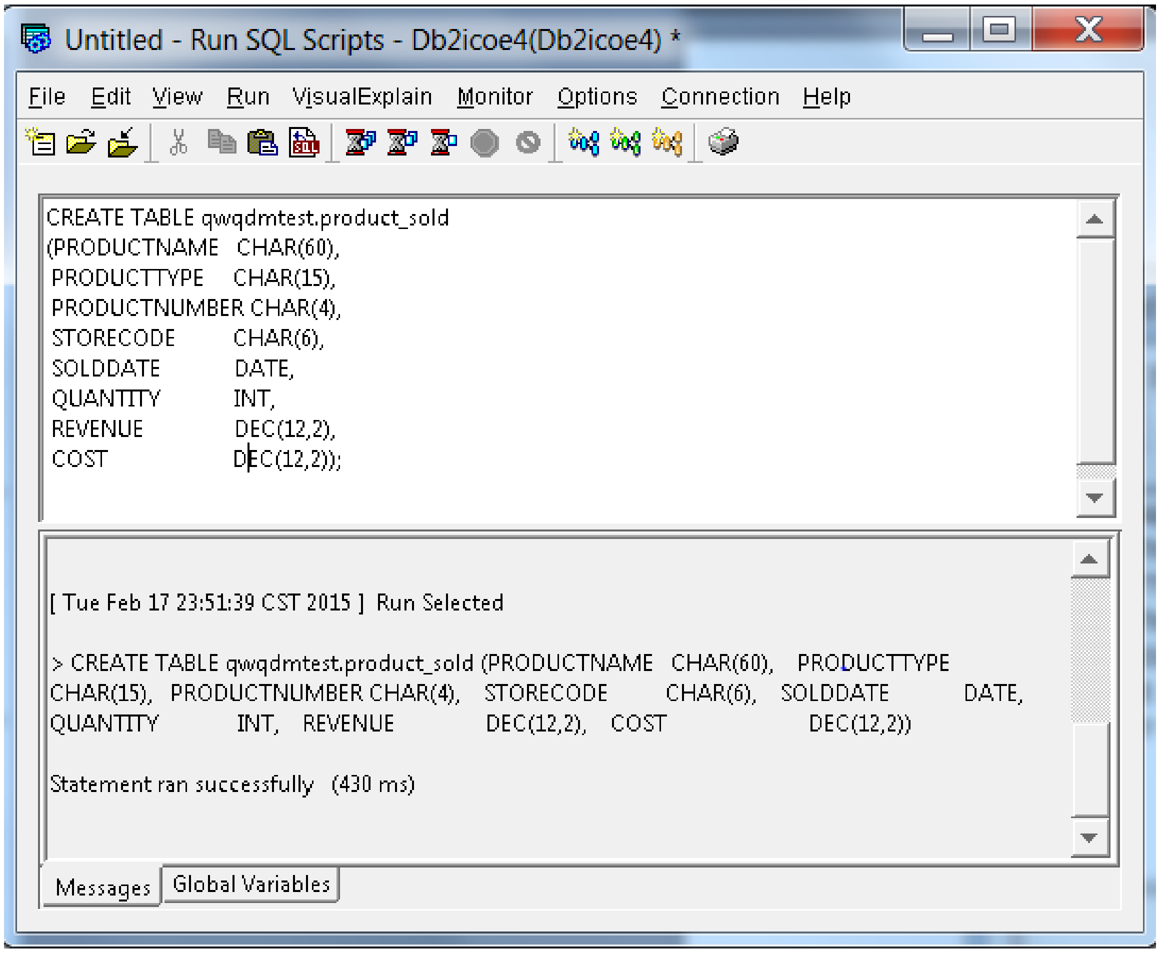
In the previous chapter we created a simple data flow to illustrate the main components of a data flow. Let’s return to the Data Flow to create a more involved example.

Preparation

Before we get started on the flow, let’s first create a table called **product_sold** into library **qwqdmtest** that will be used later in this chapter as a data target.

Using an SQL interface (System i Navigator, STRSQL…) create the following SQL table. Note that you may need to change to a system naming **qwqdmtest/product_sold** format instead of SQL naming **qwqdmtest.product_sold**.

```
CREATE TABLE qwqdmtest.product_sold
(PRODUCTNAME   CHAR(60),
 PRODUCTTYPE   CHAR(15),
 PRODUCTNUMBER CHAR(4),
 STORECODE     CHAR(6),
 SOLDDATE      DATE,
 QUANTITY      INT,
 REVENUE       DEC(12,2),
 COST          DEC(12,2));
```


Untitled - Run SQL Scripts - Db2icoe4(Db2icoe4) *
File Edit View Run VisualExplain Monitor Options Connection Help
Messages
Global Variables

```
CREATE TABLE qwqdmtest.product_sold
(PRODUCTNAME   CHAR(60),
 PRODUCTTYPE   CHAR(15),
 PRODUCTNUMBER CHAR(4),
 STORECODE     CHAR(6),
 SOLDDATE      DATE,
 QUANTITY      INT,
 REVENUE       DEC(12,2),
 COST          DEC(12,2));
```

```
[ Tue Feb 17 23:51:39 CST 2015 ]  Run Selected

> CREATE TABLE qwqdmtest.product_sold (PRODUCTNAME   CHAR(60),   PRODUCTTYPE
CHAR(15),  PRODUCTNUMBER CHAR(4),   STORECODE      CHAR(6),   SOLDDATE        DATE,
QUANTITY         INT,   REVENUE           DEC(12,2),   COST                DEC(12,2))

Statement ran successfully   (430 ms)
```

Figure 36: Create table via System i Navigator

Once the table is created, create a synonym using the DMC, searching only for the new table. Give the new synonym a name with a prefix of *flow_*

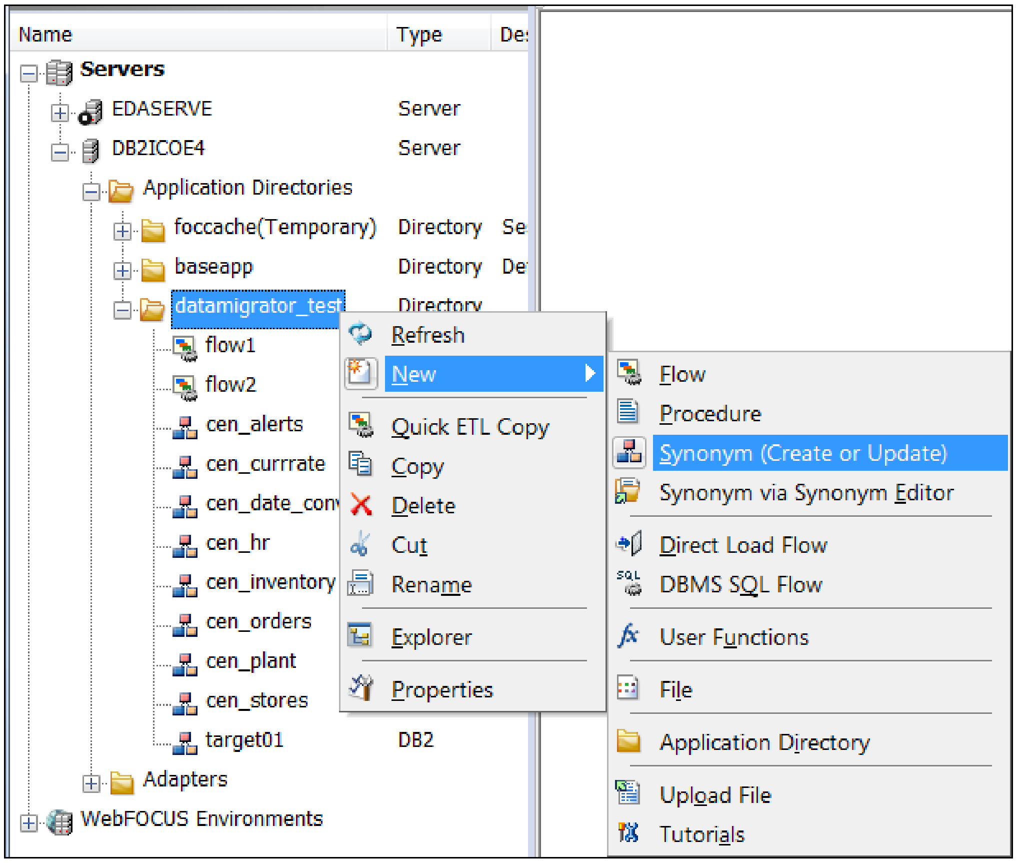


Figure 37: Create synonym over new table

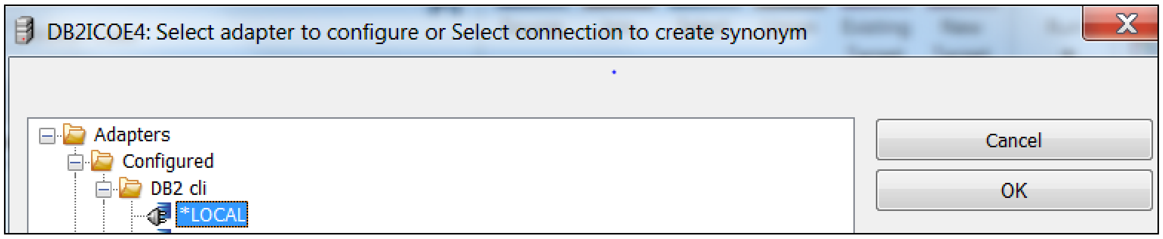
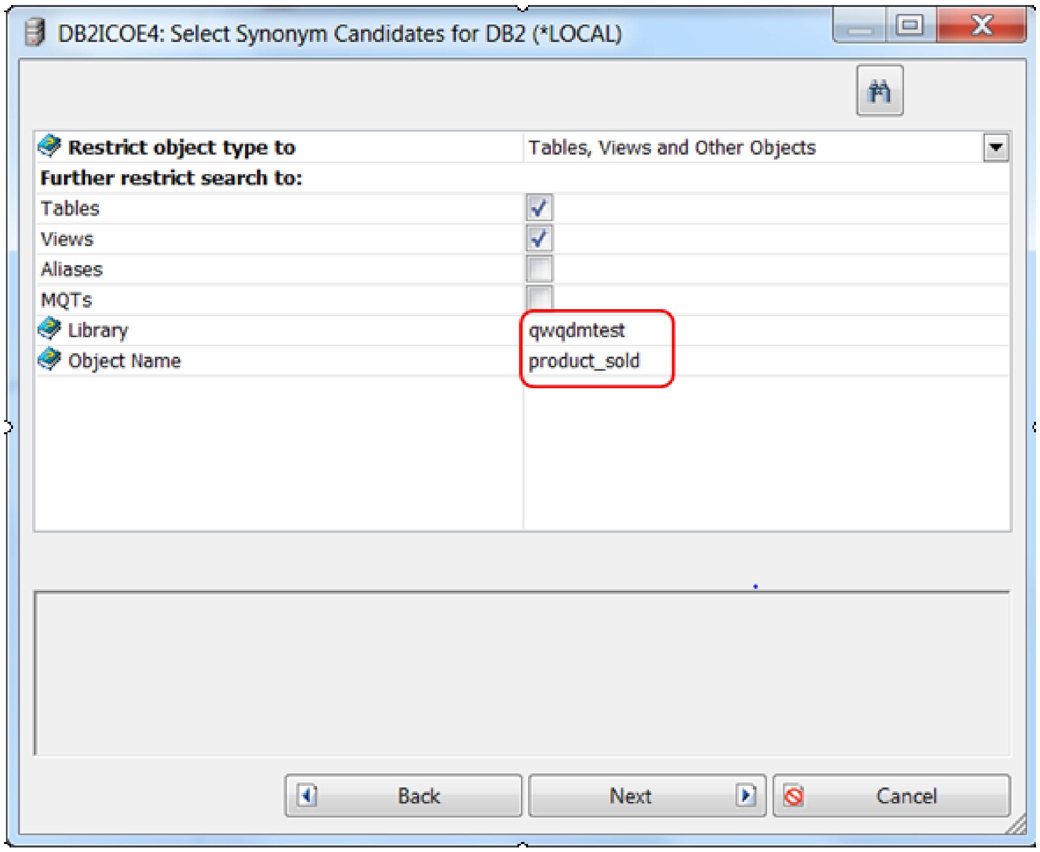
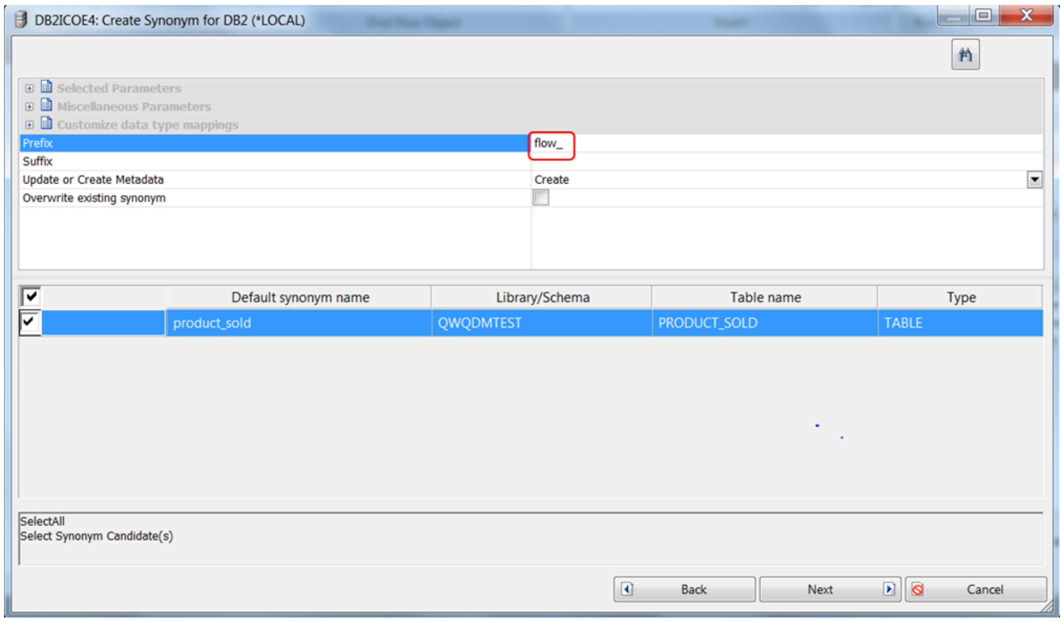

DB2ICOE4: Select adapter to configure or Select connection to create synonym
Adapters
Configured
DB2 cli
*LOCAL
Cancel
OK

Figure 38: Specify *LOCAL adapter


DB2ICOE4: Select Synonym Candidates for DB2 (*LOCAL)

Restrict object type to	Tables, Views and Other Objects
Further restrict search to:	
Tables	✓
Views	✓
Aliases	
MQTs	
Library	qwqdmtest
Object Name	product_sold

Figure 39: Search for only table qwqdmtest/product_sold


DB2ICOE4: Create Synonym for DB2 (*LOCAL)

Prefix	flow_
Suffix	
Update or Create Metadata	Create
Overwrite existing synonym	

✓	Default synonym name	Library/Schema	Table name	Type
✓	product_sold	QWQDMTEST	PRODUCT_SOLD	TABLE

SelectAll
Select Synonym Candidate(s)

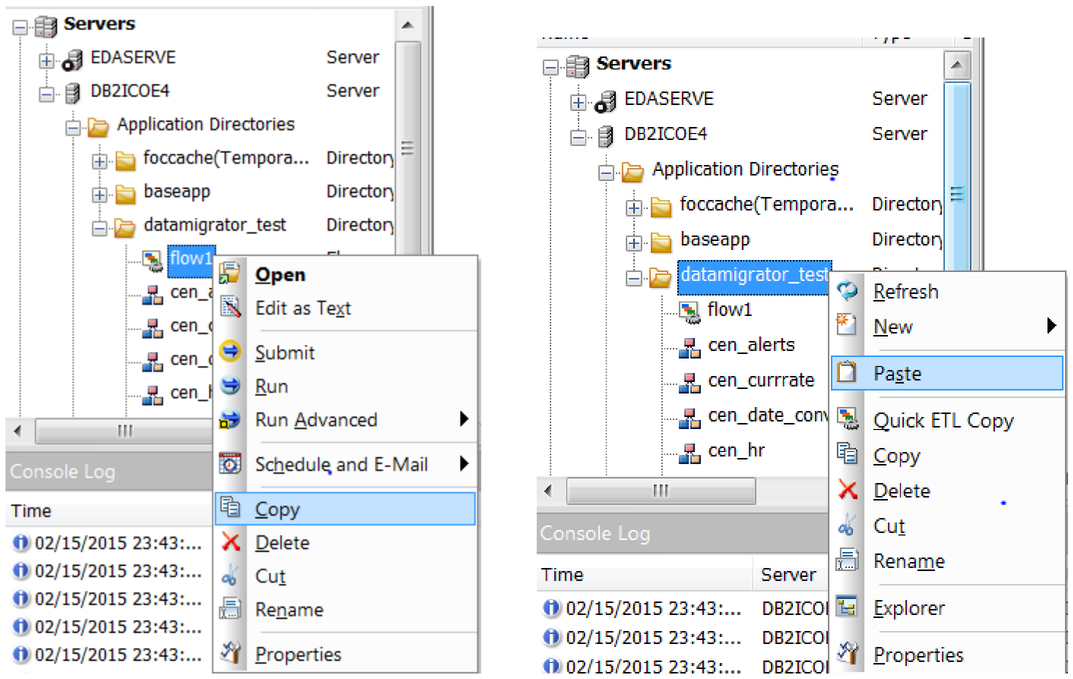
Figure 40: Specify a prefix of 'flow_'

With that data target in place, let's create the data flow.

Copying a Flow

While we could just create a new flow, this is a good chance to try out the copy flow capability.

First, copy the data flow by right clicking on the flow in the tree navigation and choose *Copy*. To paste, right click on the folder and choose *Paste*. Call the new flow **flow2**.



```
02/15/2015 23:43:...
02/15/2015 23:43:...
02/15/2015 23:43:...
02/15/2015 23:43:...
02/15/2015 23:43:...
```

Figure 41: Copy and paste to make new flow.

Open flow **flow2**.

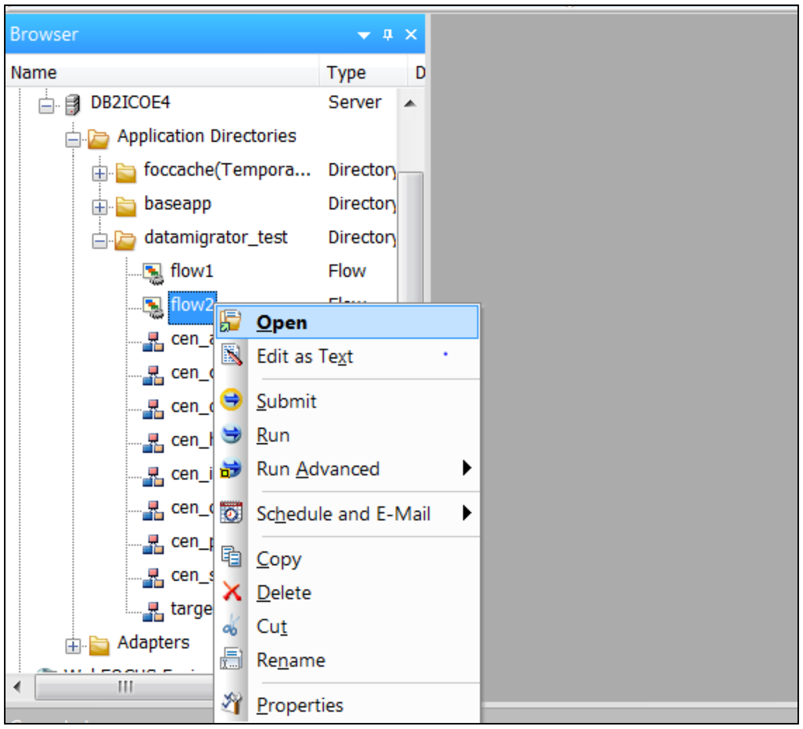


Figure 42: Opening a flow.

Flow2 starts from where we left off in the previous chapter. The Inventory data source and target are shown.

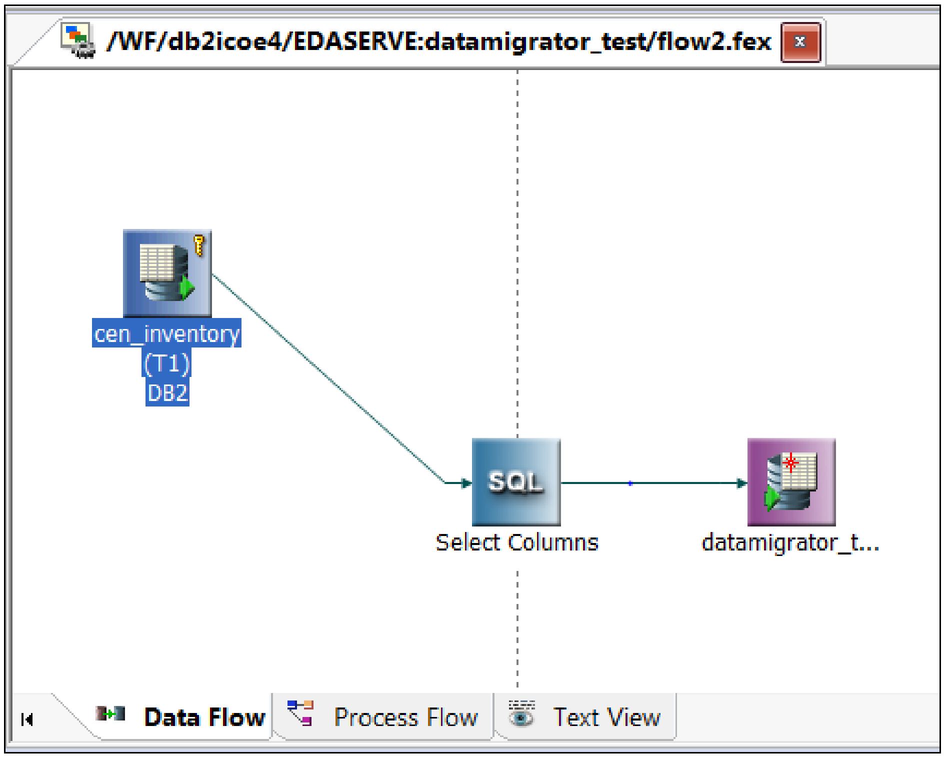
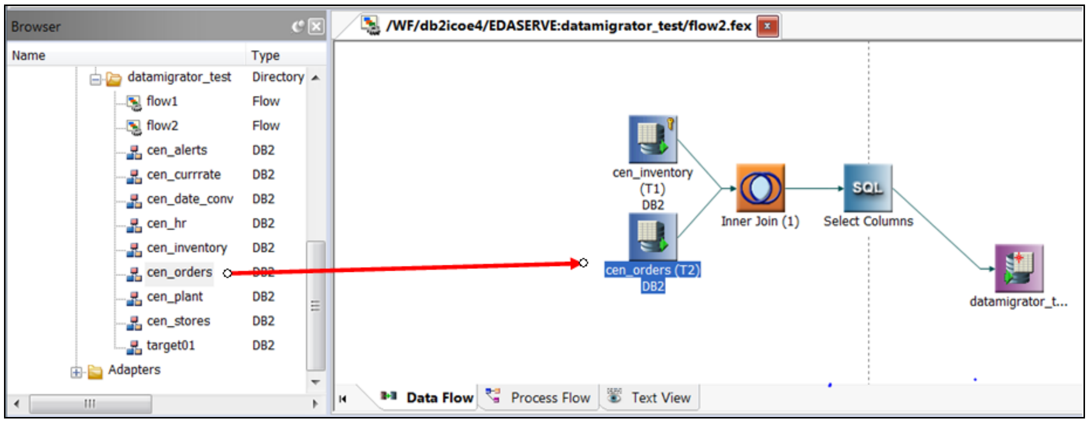


Figure 43: Starting point for flow2

Adding Joins and Transformations

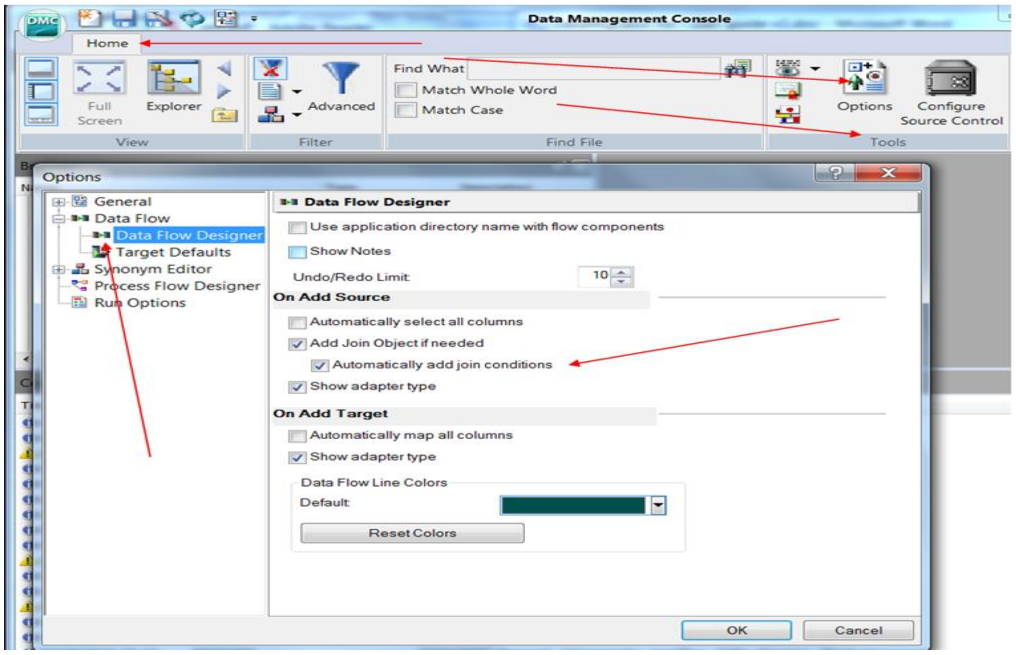
Now drag in the orders data source cen_orders. Notice how DMC automatically applies a join connector between the inventory and orders sources.



Name	Type
datamigrator_test	Directory
flow1	Flow
flow2	Flow
cen_alerts	DB2
cen_currrate	DB2
cen_date_conv	DB2
cen_hr	DB2
cen_inventory	DB2
cen_orders	DB2
cen_plant	DB2
cen_stores	DB2
target01	DB2
Adapters	

Figure 44: Adding orders table

Note: The connecting arrows between these objects are added because the option *Automatically add join conditions* is selected by default. If these connections are not appearing on your screen, go to the *Tools* menu and choose *Options*. Click the *Data Flow Designer* link and then click *Automatically add join conditions*.



Data Flow Designer

- [ ] Use application directory name with flow components
- [ ] Show Notes

Undo/Redo Limit: 10

On Add Source

- [ ] Automatically select all columns
- [✓] Add Join Object if needed
  - [✓] Automatically add join conditions
- [✓] Show adapter type

On Add Target

- [ ] Automatically map all columns
- [✓] Show adapter type

Data Flow Line Colors

Default: 🟩

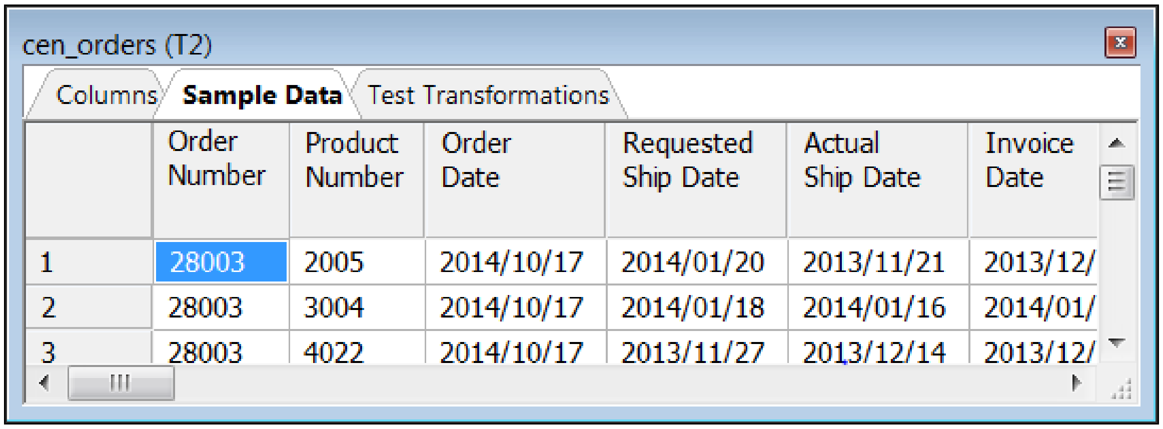
Reset Colors

Figure 45: Data Flow Designer in Options from Home tab

Let's verify the data in the **orders** table using a different method.

Right-click the **cen_orders** source object and click *Toggle*. The Columns tab shows the columns in the data source.

Click the *Sample Data* tab to verify that data is being retrieved from the orders table. The first 50 rows are retrieved.



	Order Number	Product Number	Order Date	Requested Ship Date	Actual Ship Date	Invoice Date
1	28003	2005	2014/10/17	2014/01/20	2013/11/21	2013/12/
2	28003	3004	2014/10/17	2014/01/18	2014/01/16	2014/01/
3	28003	4022	2014/10/17	2013/11/27	2013/12/14	2013/12/

Figure 46: Sample data from Toggle view

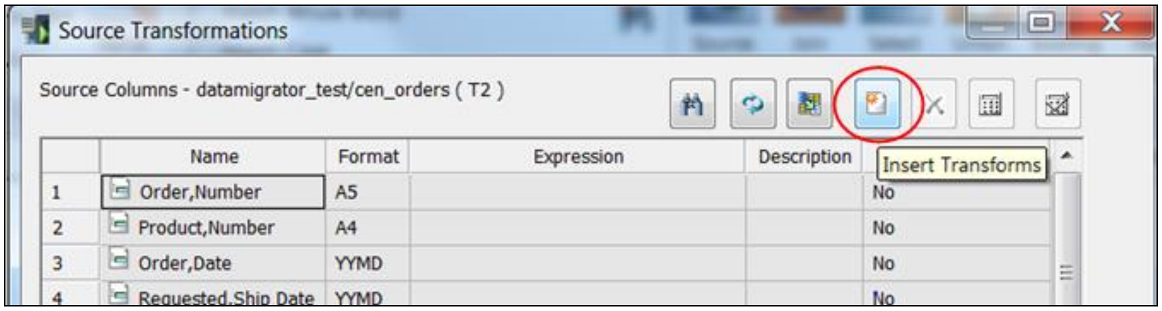
> **Note:** You can change the number of rows to retrieve from the Tools group. On the *Home* tab, in the *Tools* group, click *Options*. From the Tools dialog box, click the *Run Options* link, and then change the number in *Maximum number of rows for test reports*.

Double-click the title bar or click the **X** in the upper-right corner to close the window and return to the object view.

Next, create a virtual column in the orders data source. This is also called a *Source Transformation*. Data transformations in a source object are performed when the records are read, **before** any filtering or aggregation occurs.

The virtual column will be the computed difference in days from when an order shipped to when it was requested to be shipped.

Right click on the orders data source and choose *Source Transformations* to bring up the Source Transformations list. Choose *Insert Transforms*.



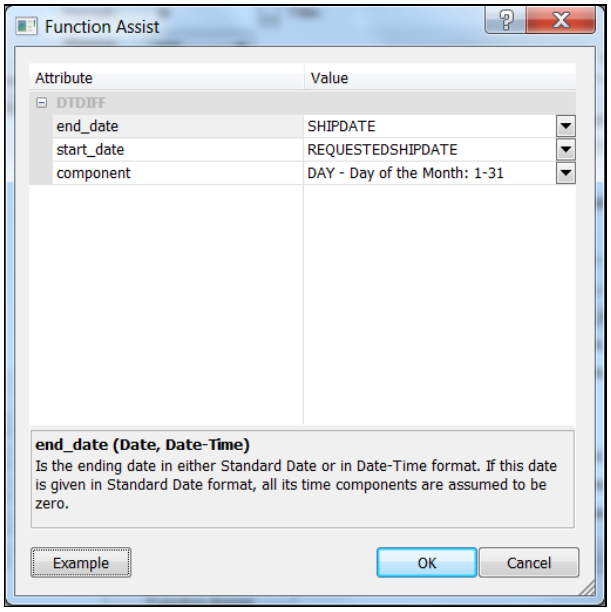
Source Columns - datamigrator_test/cen_orders (T2)

	Name	Format	Expression	Description	
1	Order,Number	A5			No
2	Product,Number	A4			No
3	Order,Date	YYMD			No
4	Requested,Ship Date	YYMD			No

Figure 47: Create virtual column

The Transformation Calculator opens.

1. In the Name box, type **SHIP_DIFFERENCE**.
2. In the *Functions* tab, expand the *Date/Date Time –Simplified* folder, double click on the DTDIFF function. This will bring up the Function Assist window.
3. For the *end_date*, use the drop down to choose column **Actual Ship Date** (SHIPDATE).
4. For the *start_date*, use the drop down to choose column **Requested Ship Date** (REQUESTEDSHIPDATE).
5. Leave *component* as DAY.



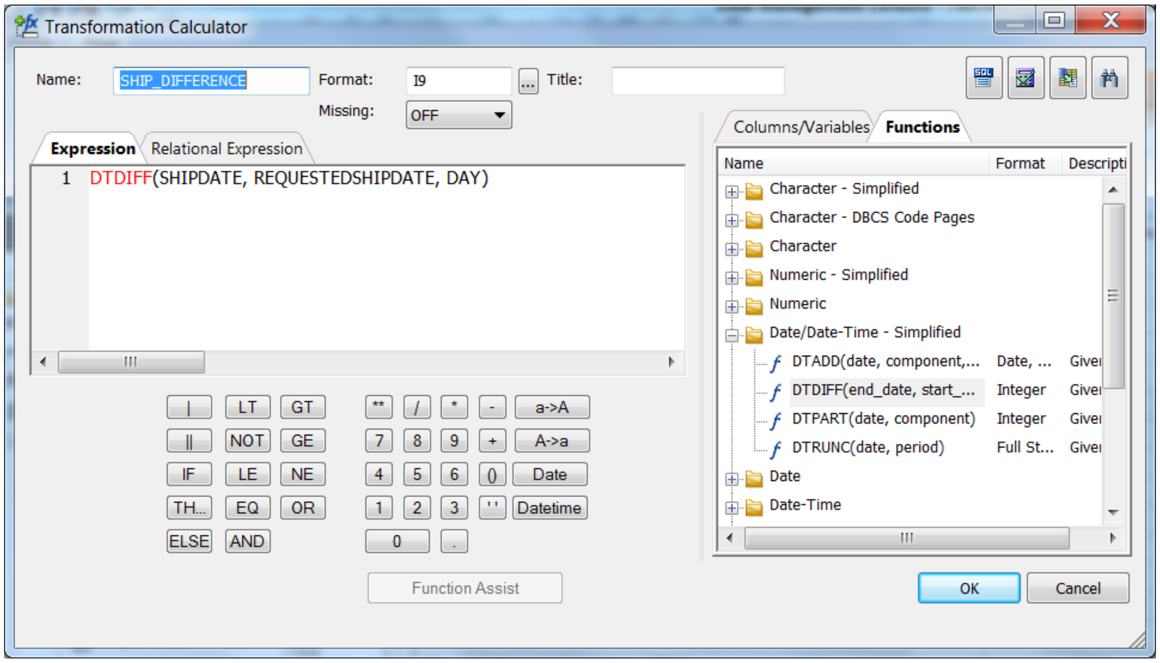
Attribute	Value
DTDIFF	
end_date	SHIPDATE
start_date	REQUESTEDSHIPDATE
component	DAY - Day of the Month: 1-31

end_date (Date, Date-Time)
Is the ending date in either Standard Date or in Date-Time format. If this date is given in Standard Date format, all its time components are assumed to be zero.

Figure 48: Function Assist window

6. Select OK.

The calculator should show the completed DTDIFF function.

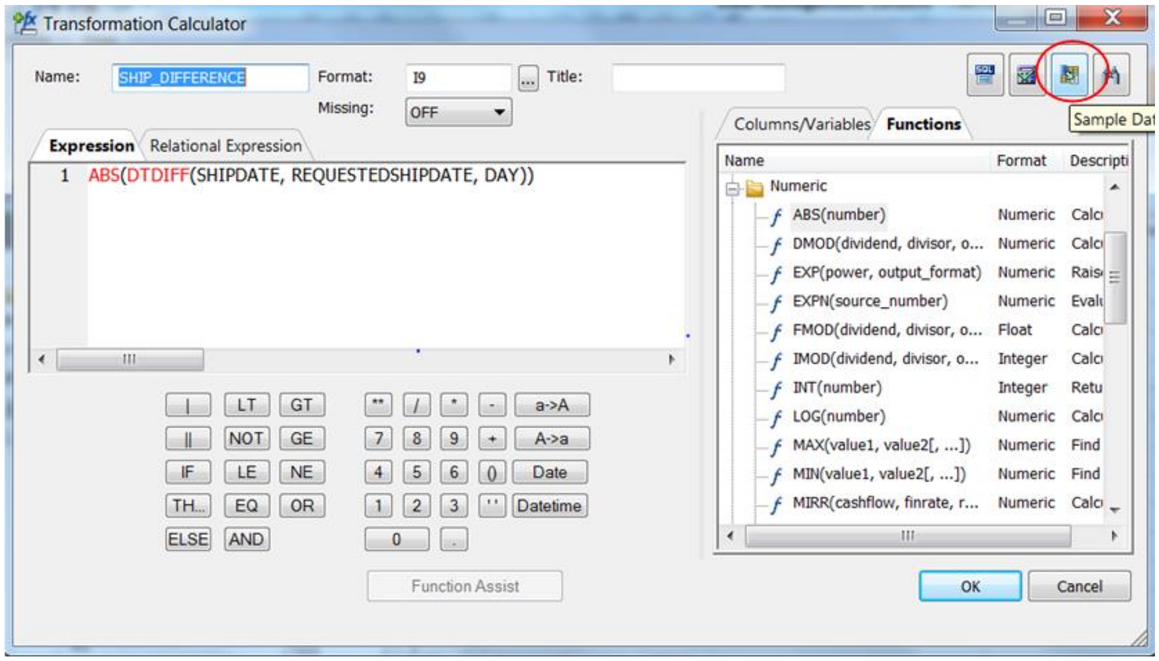


```
DTDIFF(SHIPDATE, REQUESTEDSHIPDATE, DAY)
```

Name	Format	Descripti
Character - Simplified		
Character - DBCS Code Pages		
Character		
Numeric - Simplified		
Numeric		
Date/Date-Time - Simplified		
DTADD(date, component,...	Date, ...	Giver
DTDIFF(end_date, start_...	Integer	Giver
DTPART(date, component)	Integer	Giver
DTRUNC(date, period)	Full St...	Giver
Date		
Date-Time		

Figure 49: SHIP_DIFFERENCE virtual column definition using DTDIFF

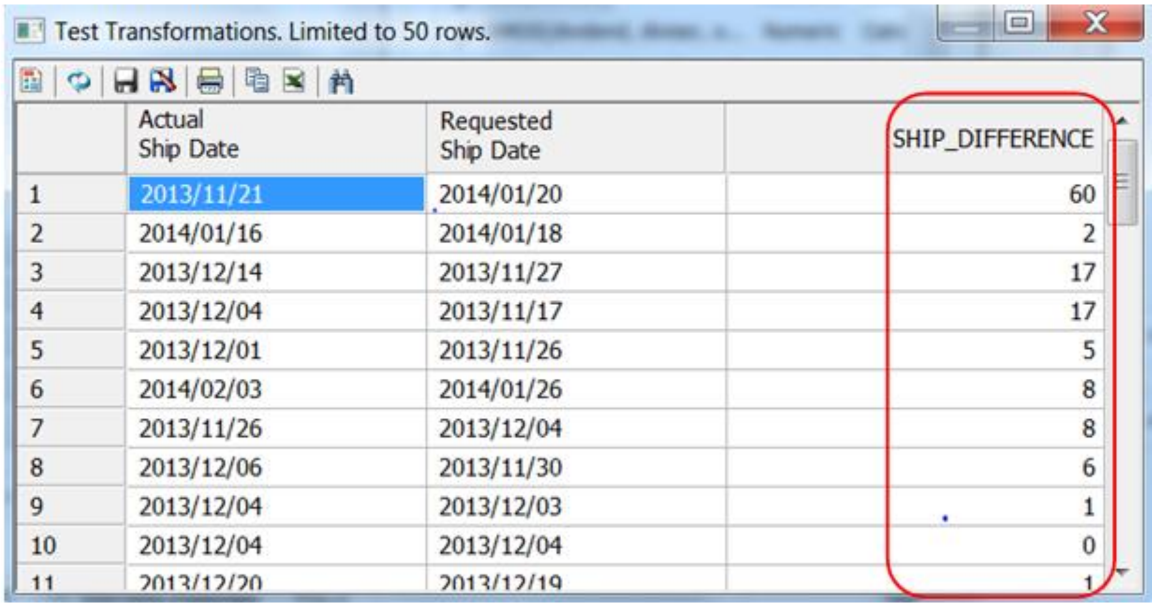
We are not quite done. We are only interested in the absolute difference between the actual and the requested date. Therefore, let’s wrap the result in an ABS function. Add a **ABS(** at the beginning and a closing **)** at the end of the Expression. Then click the Sample Data button on the upper right to make sure the expression is valid.



```
ABS(DTDIFF(SHIPDATE, REQUESTEDSHIPDATE, DAY))
```

Name	Format	Descripti
Numeric		
ABS(number)	Numeric	Calc
DMOD(dividend, divisor, o...	Numeric	Calc
EXP(power, output_format)	Numeric	Rais
EXPN(source_number)	Numeric	Evalu
FMOD(dividend, divisor, o...	Float	Calc
IMOD(dividend, divisor, o...	Integer	Calc
INT(number)	Integer	Retu
LOG(number)	Numeric	Calc
MAX(value1, value2[, ...])	Numeric	Find
MIN(value1, value2[, ...])	Numeric	Find
MIRR(cashflow, finrate, r...	Numeric	Calc

Figure 50: ABS function



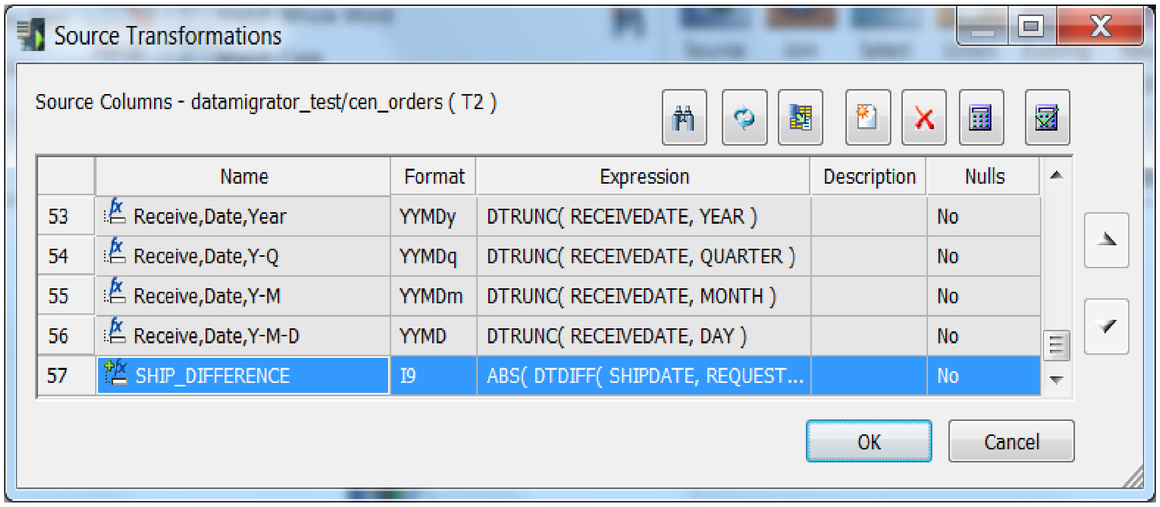
Test Transformations. Limited to 50 rows.

	Actual Ship Date	Requested Ship Date	SHIP_DIFFERENCE
1	2013/11/21	2014/01/20	60
2	2014/01/16	2014/01/18	2
3	2013/12/14	2013/11/27	17
4	2013/12/04	2013/11/17	17
5	2013/12/01	2013/11/26	5
6	2014/02/03	2014/01/26	8
7	2013/11/26	2013/12/04	8
8	2013/12/06	2013/11/30	6
9	2013/12/04	2013/12/03	1
10	2013/12/04	2013/12/04	0
11	2013/12/20	2013/12/19	1

Figure 51: Sample Data showing calculated ship date difference

Close the *Sample Data* window, then click OK on the *Transformation Calculator*. **SHIP_DIFFERENCE** should now show up at the bottom of the Source Transformations list.

> **Note**: Virtual columns are identified by the ***fx*** symbol. Putting a virtual column in a synonym (rather than a flow) is a useful strategy when you expect to use the same synonym with more than one flow.



Source Transformations

Source Columns - datamigrator_test/cen_orders (T2)

	Name	Format	Expression	Description	Nulls
53	Receive,Date,Year	YYMDy	DTRUNC(RECEIVEDATE, YEAR)		No
54	Receive,Date,Y-Q	YYMDq	DTRUNC(RECEIVEDATE, QUARTER)		No
55	Receive,Date,Y-M	YYMDm	DTRUNC(RECEIVEDATE, MONTH)		No
56	Receive,Date,Y-M-D	YYMD	DTRUNC(RECEIVEDATE, DAY)		No
57	SHIP_DIFFERENCE	I9	ABS(DTDIFF(SHIPDATE, REQUEST...		No

Figure 52: SHIP_DIFFERENCE in the Source Transformations list.

Click OK on the *Source Transformation* window.

Now let us consider the join object, which was added automatically when you selected the second data source. You will need to specify properties for the join.

By default, an inner join is created. An inner join extracts those rows that appear in both tables. You will base the join on an equality condition between two fields, one in each data source. The use of an equality condition is also called an equi-join.

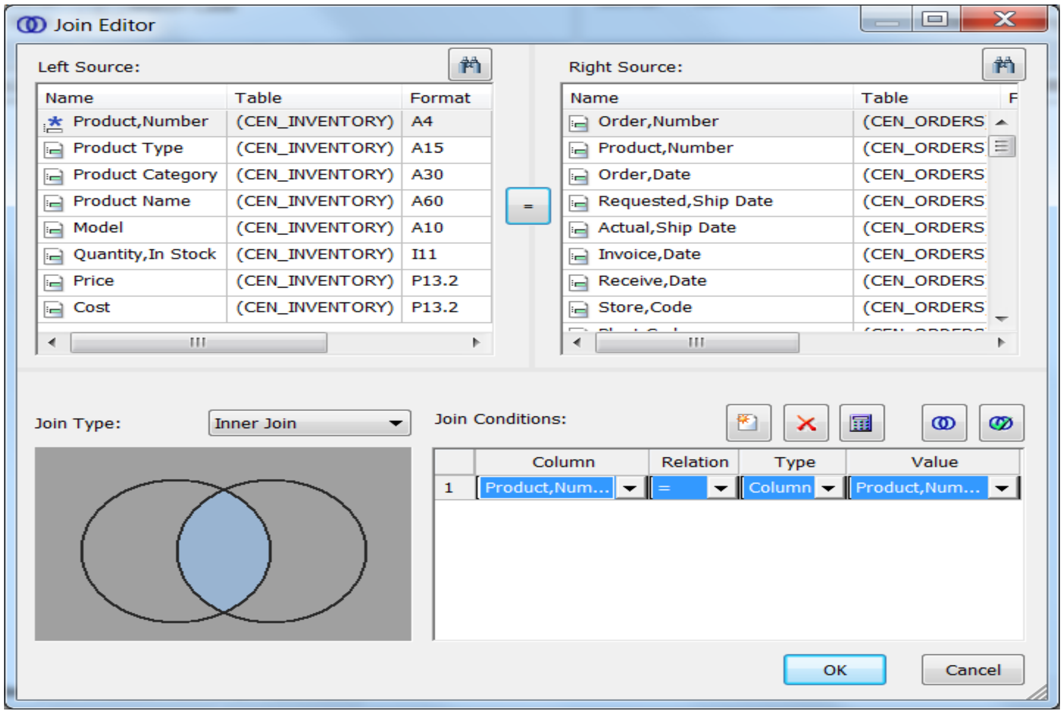
> **Note:** DataMigrator for i supports multiple joins, joins based on conditions other than equalities, and joins that are modified by calculations, such as substrings or concatenations. A Join Calculator is available to assist you.

Right-click the join object and click **Join Editor**. The Join Editor window opens.

The join must be based on columns in each of the joined data sources. Notice that **Product Number** is in both Left and Right Source Columns lists. The join of **Product Number** between the data sources appears in the Expression field of the Join Conditions list. For our purposes, the default join on **Product Number** is sufficient.

> **Note:** Once again, the automatic join condition is in effect because the option *Automatically add Join conditions* is selected by default. If this is not the case, on the Home tab, in the Tools group, click *Options*. In the Options dialog box, click the *Data Flow Designer* link and select *Automatically add Join conditions*.

The inner join relationship is reflected in the Expression box. It is represented graphically by the overlapping area in the Join Type diagram, as shown in the following image.



Left Source:

Name	Table	Format
Product,Number	(CEN_INVENTORY)	A4
Product Type	(CEN_INVENTORY)	A15
Product Category	(CEN_INVENTORY)	A30
Product Name	(CEN_INVENTORY)	A60
Model	(CEN_INVENTORY)	A10
Quantity,In Stock	(CEN_INVENTORY)	I11
Price	(CEN_INVENTORY)	P13.2
Cost	(CEN_INVENTORY)	P13.2

Right Source:

Name	Table	F
Order,Number	(CEN_ORDERS	
Product,Number	(CEN_ORDERS	
Order,Date	(CEN_ORDERS	
Requested,Ship Date	(CEN_ORDERS	
Actual,Ship Date	(CEN_ORDERS	
Invoice,Date	(CEN_ORDERS	
Receive,Date	(CEN_ORDERS	
Store,Code	(CEN_ORDERS	

Join Conditions:

	Column	Relation	Type	Value
1	Product,Num...	=	Column	Product,Num...

Figure 53: Inner join relationship on Product Number

Note: If you want to change the type to a right or left outer join, you can simply click the left or right circle. Try that now if you like, and then return to the inner join position.

Click OK to close the *Join Editor* window.

Selecting Columns and Transformations

We are now ready to select the columns of data to load into the data targets. There are a variety of operations on the selected columns.

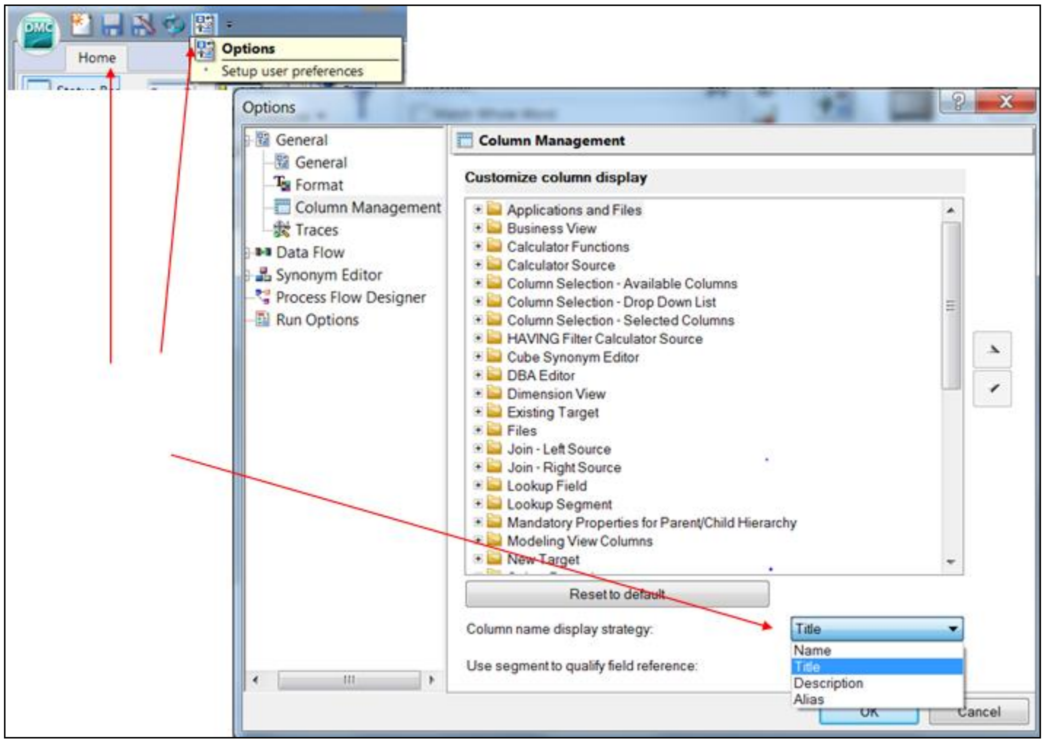
Right-click the SQL object and notice the options on the menu.

- *Toggle***.** Opens an information window, as you saw earlier.
- *SQL Statement***.** Displays the SQL code. Right now it simply reflects the join. It will be more interesting in a little while.
- *WHERE Filter***.** Provides a calculator in which you can create an expression that limits record selection. For example, you might only want to retrieve records for a certain year.
- *HAVING Filter***.** Provides a calculator in which you can create an expression that limits retrieval based on aggregated values after a GROUP BY. This option only appears when the Column Selection includes a GROUP BY.
- *Sort.* Provides a dialog box in which you can control the order in which data is retrieved.
- *Column Selection***.** Opens a window in which you select the columns you want to include in your data target, and specify a variety of data retrieval requirements, as you will do in the following steps.
- *Add***.** Allows you to add additional sources, selects, join, and union objects.
- *Properties***.** Opens a property panel that shows statistics for the select statement.
- *Error Details***.** This option only appears if there is an error in the columns or filters. When selected, it and a dialog box opens to explain the error.

Click the *Save* button to save our data flow up to this point.

Right-click the *SQL* column object and click *Column Selection.*

Note: DMC options control whether title, name or description is shown as the main aspect for columns. If your screen does not show the column attribute you need or that is shown throughout this user guide, change the option using the *Options* button from the main DMC screen under the *Home* tab.



Column Management

Customize column display

- Applications and Files
- Business View
- Calculator Functions
- Calculator Source
- Column Selection - Available Columns
- Column Selection - Drop Down List
- Column Selection - Selected Columns
- HAVING Filter Calculator Source
- Cube Synonym Editor
- DBA Editor
- Dimension View
- Existing Target
- Files
- Join - Left Source
- Join - Right Source
- Lookup Field
- Lookup Segment
- Mandatory Properties for Parent/Child Hierarchy
- Modeling View Columns
- New Target

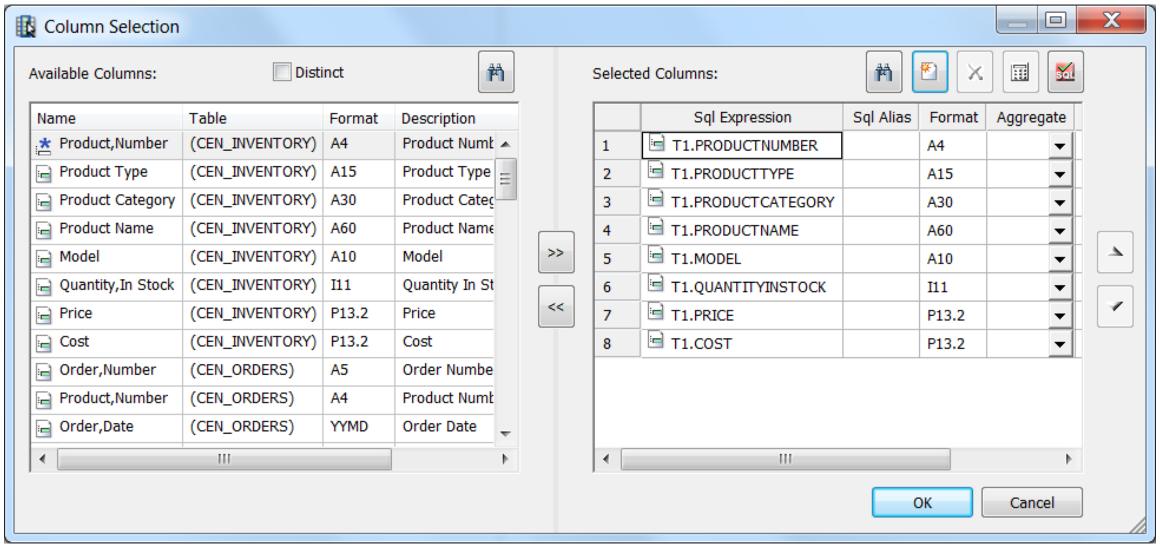
Column name display strategy: Title

Name / Title / Description / Alias

Use segment to qualify field reference:

Figure 54: Changing column naming

Because this was a copy of the first data flow, the columns from inventory are already selected as carryover from the first data flow.



Available Columns:

Name	Table	Format	Description
Product,Number	(CEN_INVENTORY)	A4	Product Numt
Product Type	(CEN_INVENTORY)	A15	Product Type
Product Category	(CEN_INVENTORY)	A30	Product Cate(
Product Name	(CEN_INVENTORY)	A60	Product Name
Model	(CEN_INVENTORY)	A10	Model
Quantity,In Stock	(CEN_INVENTORY)	I11	Quantity In St
Price	(CEN_INVENTORY)	P13.2	Price
Cost	(CEN_INVENTORY)	P13.2	Cost
Order,Number	(CEN_ORDERS)	A5	Order Numbe
Product,Number	(CEN_ORDERS)	A4	Product Numt
Order,Date	(CEN_ORDERS)	YYMD	Order Date

Selected Columns:

	Sql Expression	Sql Alias	Format	Aggregate
1	T1.PRODUCTNUMBER		A4	
2	T1.PRODUCTTYPE		A15	
3	T1.PRODUCTCATEGORY		A30	
4	T1.PRODUCTNAME		A60	
5	T1.MODEL		A10	
6	T1.QUANTITYINSTOCK		I11	
7	T1.PRICE		P13.2	
8	T1.COST		P13.2	

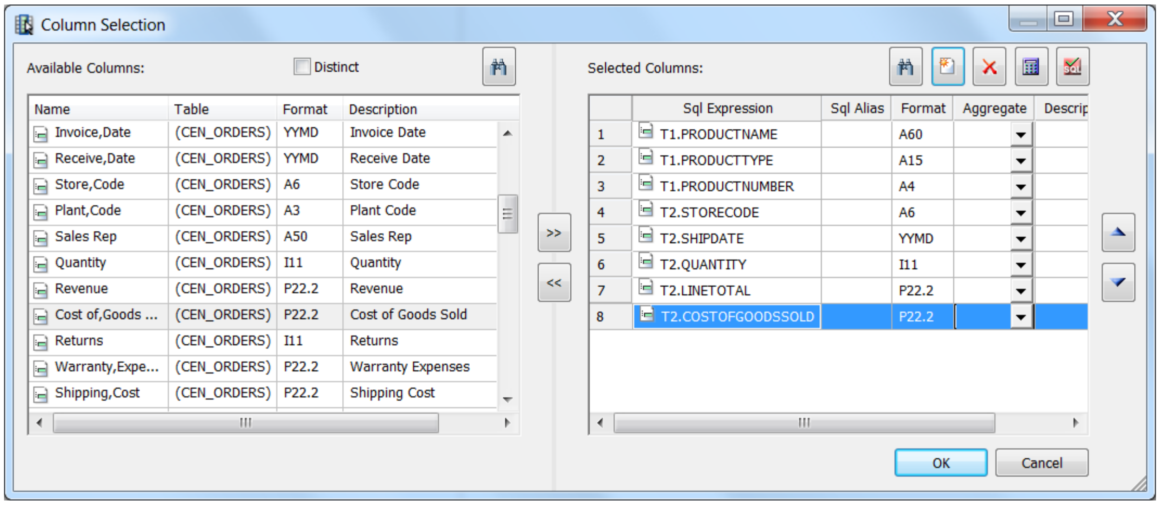
Figure 55: Initial column selection list

We want to create a flow to track product and store information by month. We also want to track the average days that the actual ship date differs from the requested ship date.

We will be selecting the following columns:

- PRODUCTNAME
- PRODUCTTYPE
- PRODUCTNUMBER
- STORECODE
- SHIPDATE
- QUANTITY
- LINETOTAL
- COSTOFGOODSSOLD

Using the >> and << buttons in the middle of the window, move columns into and out of the Selected Columns list. To move a column into the Selected Columns, highlight the column in the *Available Columns* list and click the >> button. To remove a *Selected Column*, highlight it and click the << button. Once the columns are added, move them around with the up and down arrows on the right until they are ordered as shown.



Name	Table	Format	Description
Invoice,Date	(CEN_ORDERS)	YYMD	Invoice Date
Receive,Date	(CEN_ORDERS)	YYMD	Receive Date
Store,Code	(CEN_ORDERS)	A6	Store Code
Plant,Code	(CEN_ORDERS)	A3	Plant Code
Sales Rep	(CEN_ORDERS)	A50	Sales Rep
Quantity	(CEN_ORDERS)	I11	Quantity
Revenue	(CEN_ORDERS)	P22.2	Revenue
Cost of,Goods ...	(CEN_ORDERS)	P22.2	Cost of Goods Sold
Returns	(CEN_ORDERS)	I11	Returns
Warranty,Expe...	(CEN_ORDERS)	P22.2	Warranty Expenses
Shipping,Cost	(CEN_ORDERS)	P22.2	Shipping Cost

	Sql Expression	Sql Alias	Format	Aggregate	Descrip
1	T1.PRODUCTNAME		A60		
2	T1.PRODUCTTYPE		A15		
3	T1.PRODUCTNUMBER		A4		
4	T2.STORECODE		A6		
5	T2.SHIPDATE		YYMD		
6	T2.QUANTITY		I11		
7	T2.LINETOTAL		P22.2		
8	T2.COSTOFGOODSSOLD		P22.2		

Figure 56: Selected columns

Note: make sure to select **Quantity**, **not** Quantity in Stock!

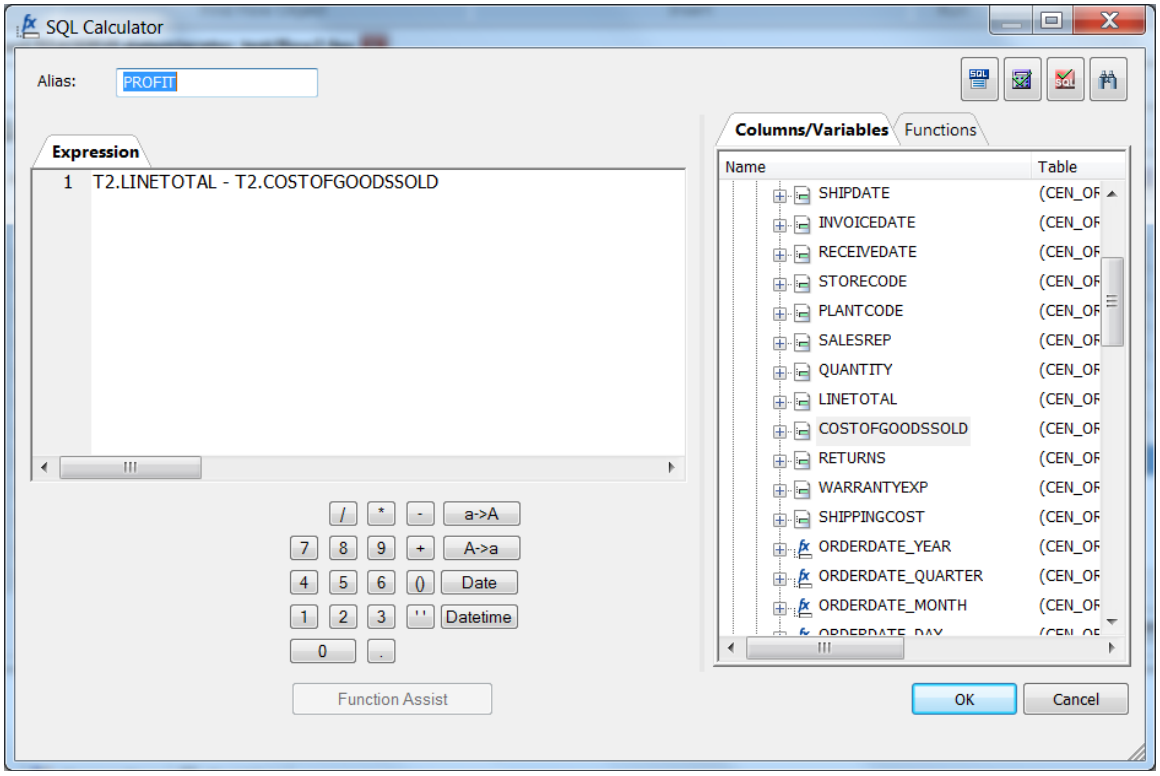
Next, we want to calculate the profit.

This transformation can be done using an SQL calculation to create the column. To open the SQL Calculator, click the *Insert Columns* button above the Selected Columns list.

- In the Alias box, enter **PROFIT**.
- Ensure that the Columns/Variables tab is selected. Then, under CEN_ORDERS, double-click **T2.LINETOTAL** in the tree. The column appears in the Expression box.
- Click the subtraction sign (-) on the calculator keypad.

- Double-click **T2.COSTOFGOODSSOLD** in the tree to complete the expression.

The SQL Calculator should look like the image below:



```
T2.LINETOTAL - T2.COSTOFGOODSSOLD
```

Name	Table
SHIPDATE	(CEN_OF
INVOICEDATE	(CEN_OF
RECEIVEDATE	(CEN_OF
STORECODE	(CEN_OF
PLANTCODE	(CEN_OF
SALESREP	(CEN_OF
QUANTITY	(CEN_OF
LINETOTAL	(CEN_OF
COSTOFGOODSSOLD	(CEN_OF
RETURNS	(CEN_OF
WARRANTYEXP	(CEN_OF
SHIPPINGCOST	(CEN_OF
ORDERDATE_YEAR	(CEN_OF
ORDERDATE_QUARTER	(CEN_OF
ORDERDATE_MONTH	(CEN_OF

Figure 57: Profit computation

Click OK to close the *SQL Calculator* and return to the *Column Selection* window.

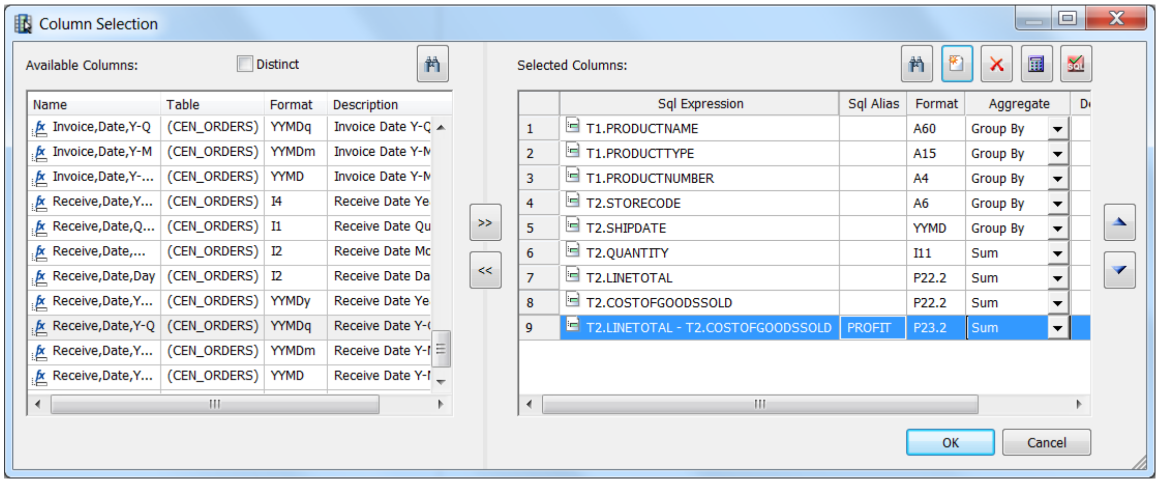
Notice that the expression has been added to the bottom of the Selected Columns list. The expression is in the first column and the alias you assigned is in the second column.

At this point we want to aggregate orders by date. We want to group by dimension columns and summarize measures. We would like total revenue, total cost, total profit and average ship date difference by date.

To do this, we must first aggregate on the key columns, and then on the year/month column. This is done by adding the *Group* attribute to each column.

Under *Selected Columns*, click **T1.PRODUCTNAME,** then select *Group By* from the drop-down menu in the *Aggregate* column. Repeat this step for **T1.PRODUCTTYPE** , **T1.PRODUCTNUMBER**, **T2.STORE_CODE** and for **T2.SHIPDATE**.

Next, sum the measures: **T2.QUANTITY**, **T2.LINETOTAL, T2.COSTOFGOODSSOLD**, and **PROFIT**. You can multi-select them. Under *Selected Columns*, press the Ctrl key and click the measures. Choose *Sum* from the drop-down menu in the *Aggregate* column. Sum is applied to all four columns, as shown in the following image.



Column Selection

Available Columns: Distinct

Name	Table	Format	Description
Invoice,Date,Y-Q	(CEN_ORDERS)	YYMDq	Invoice Date Y-Q
Invoice,Date,Y-M	(CEN_ORDERS)	YYMDm	Invoice Date Y-M
Invoice,Date,Y-...	(CEN_ORDERS)	YYMD	Invoice Date Y-M
Receive,Date,Y...	(CEN_ORDERS)	I4	Receive Date Ye
Receive,Date,Q...	(CEN_ORDERS)	I1	Receive Date Qu
Receive,Date,...	(CEN_ORDERS)	I2	Receive Date Mc
Receive,Date,Day	(CEN_ORDERS)	I2	Receive Date Da
Receive,Date,Y...	(CEN_ORDERS)	YYMDy	Receive Date Ye
Receive,Date,Y-Q	(CEN_ORDERS)	YYMDq	Receive Date Y-(
Receive,Date,Y...	(CEN_ORDERS)	YYMDm	Receive Date Y-I
Receive,Date,Y...	(CEN_ORDERS)	YYMD	Receive Date Y-I

Selected Columns:

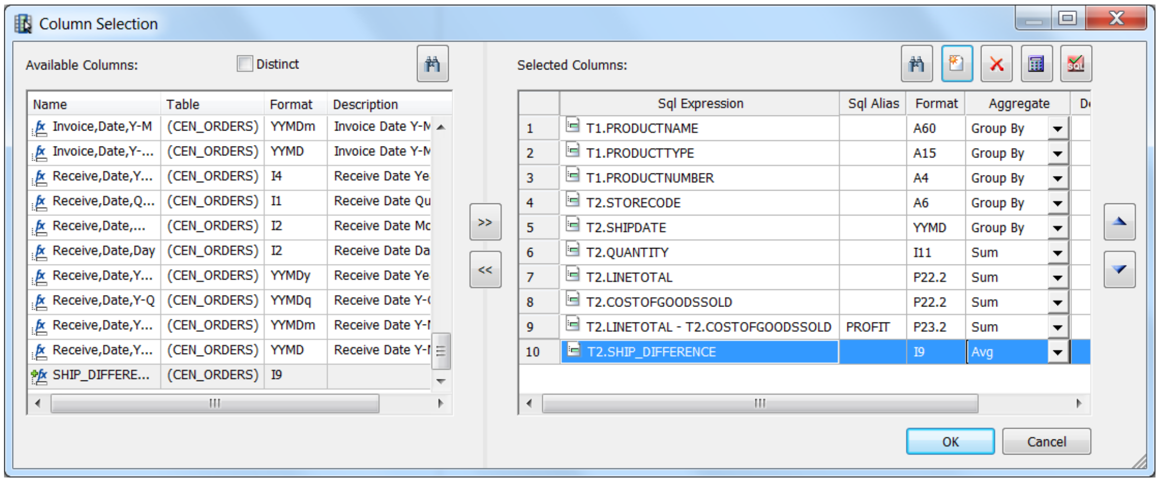
	Sql Expression	Sql Alias	Format	Aggregate
1	T1.PRODUCTNAME		A60	Group By
2	T1.PRODUCTTYPE		A15	Group By
3	T1.PRODUCTNUMBER		A4	Group By
4	T2.STORECODE		A6	Group By
5	T2.SHIPDATE		YYMD	Group By
6	T2.QUANTITY		I11	Sum
7	T2.LINETOTAL		P22.2	Sum
8	T2.COSTOFGOODSSOLD		P22.2	Sum
9	T2.LINETOTAL - T2.COSTOFGOODSSOLD	PROFIT	P23.2	Sum

OK Cancel

Figure 58: Aggregated columns

As you clicked and aggregated each column down the list, you probably noticed that the remaining columns turned red. This was an indication that, at that point, the combination of selected columns was not valid. If one selected column is aggregated, all selected columns must be aggregated. Once all columns were aggregated, the red disappeared.

Notice that we missed one selected column, the average ship date difference. Go back and pull that in now. Unlike the other measures, we want the average, not sum. Pull in the **T2.SHIP_DIFFERENCE** column and select *Avg* from the drop down. Now we have all the selected columns we want.



Column Selection

Available Columns: Distinct

Name	Table	Format	Description
Invoice,Date,Y-M	(CEN_ORDERS)	YYMDm	Invoice Date Y-M
Invoice,Date,Y-...	(CEN_ORDERS)	YYMD	Invoice Date Y-M
Receive,Date,Y...	(CEN_ORDERS)	I4	Receive Date Ye
Receive,Date,Q...	(CEN_ORDERS)	I1	Receive Date Qu
Receive,Date,...	(CEN_ORDERS)	I2	Receive Date Mc
Receive,Date,Day	(CEN_ORDERS)	I2	Receive Date Da
Receive,Date,Y...	(CEN_ORDERS)	YYMDy	Receive Date Ye
Receive,Date,Y-Q	(CEN_ORDERS)	YYMDq	Receive Date Y-(
Receive,Date,Y...	(CEN_ORDERS)	YYMDm	Receive Date Y-I
Receive,Date,Y...	(CEN_ORDERS)	YYMD	Receive Date Y-I
SHIP_DIFFERE...	(CEN_ORDERS)	I9	

Selected Columns:

	Sql Expression	Sql Alias	Format	Aggregate
1	T1.PRODUCTNAME		A60	Group By
2	T1.PRODUCTTYPE		A15	Group By
3	T1.PRODUCTNUMBER		A4	Group By
4	T2.STORECODE		A6	Group By
5	T2.SHIPDATE		YYMD	Group By
6	T2.QUANTITY		I11	Sum
7	T2.LINETOTAL		P22.2	Sum
8	T2.COSTOFGOODSSOLD		P22.2	Sum
9	T2.LINETOTAL - T2.COSTOFGOODSSOLD	PROFIT	P23.2	Sum
10	T2.SHIP_DIFFERENCE		I9	Avg

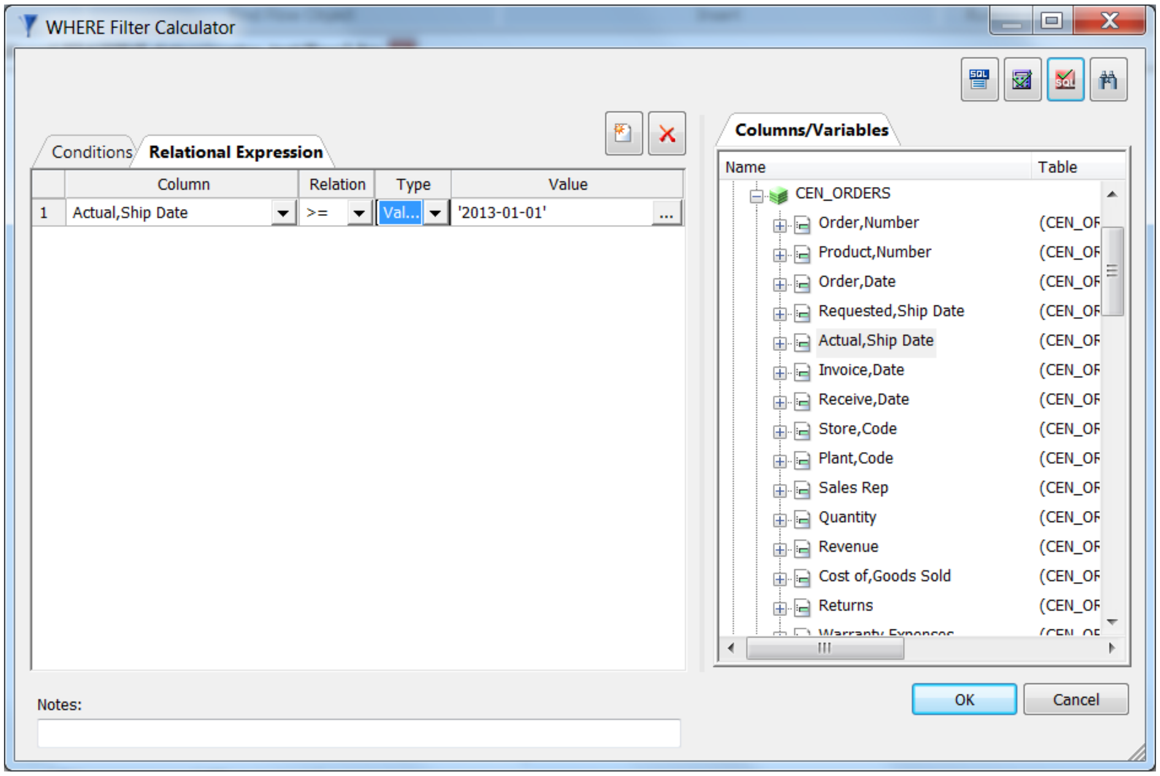
OK Cancel

Figure 59: Aggregated columns and SHIP_DIFFERENCE

Click OK to close the *Column Selection* window and return to the object view in the *Data Flow* tab.

We have identified the columns to extract from the source data, but the data in the source goes back several years and only the last few years are interesting. Therefore, we need to define a selection criterion to limit the retrieval to the appropriate years.

- Right-click the *SQL* object again, and this time click *WHERE Filter*. The *WHERE Filter Calculator* opens to assist you in constructing the expression. Notice that this calculator is suitable for creating a wide range of selection criteria.
- As you can see, the columns here are represented by the same icons used in the *Selected Columns* dialog box. You can filter on real or virtual columns.
- To limit record retrieval to the time period beginning in the year 2013:
    a. Double-click **Actual Ship Date** under **cen_orders** in the Column list. **Actual Ship Date** is displayed in the Expression box.
    b. Click >= (greater than or equal to) in the calculator pad below the Expression window.
    c. Specify *Type* of *Value*
    d. Type in a date, with quotes, of ‘2013-01-01’
- The expression in the WHERE Filter Calculator should look like the following image:



	Column	Relation	Type	Value
1	Actual,Ship Date	>=	Val...	'2013-01-01'

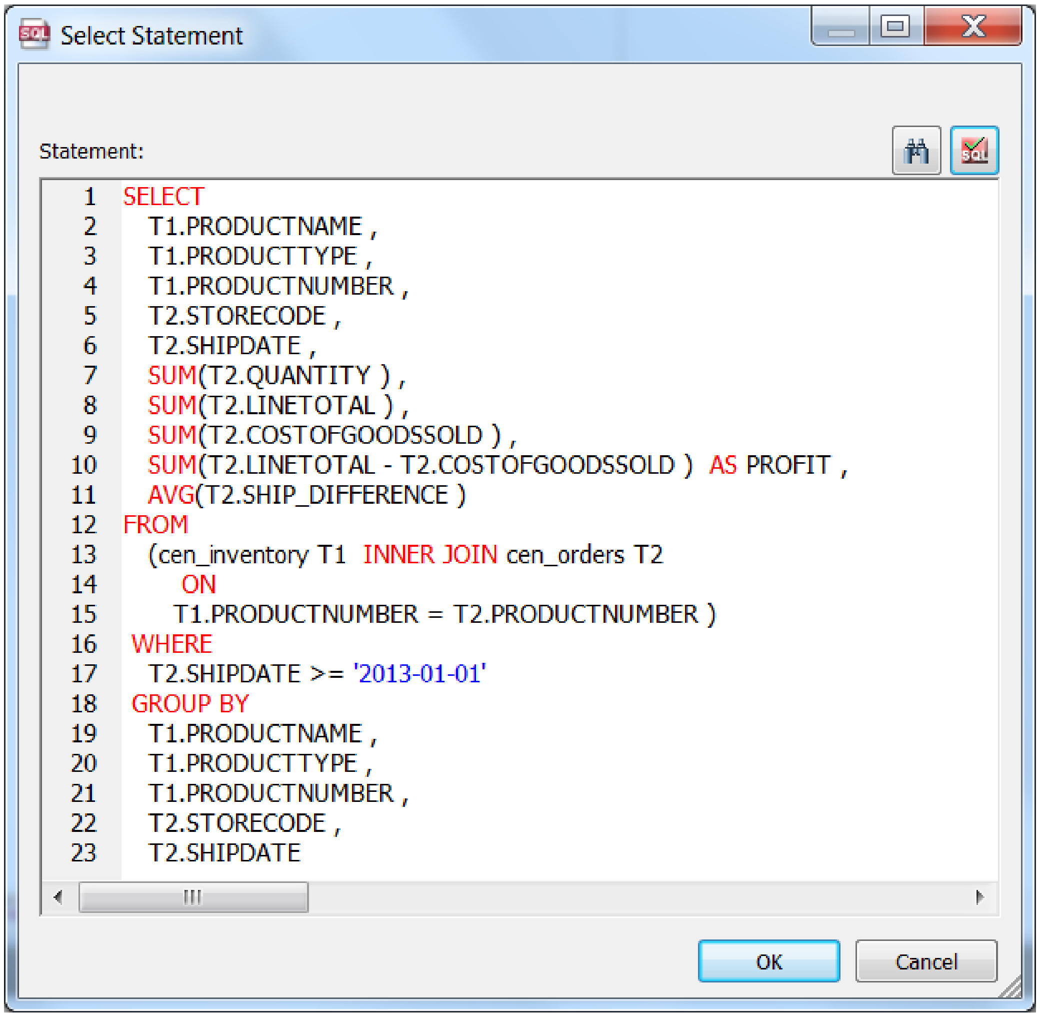
Name	Table
CEN_ORDERS	
Order,Number	(CEN_OF
Product,Number	(CEN_OF
Order,Date	(CEN_OF
Requested,Ship Date	(CEN_OF
Actual,Ship Date	(CEN_OF
Invoice,Date	(CEN_OF
Receive,Date	(CEN_OF
Store,Code	(CEN_OF
Plant,Code	(CEN_OF
Sales Rep	(CEN_OF
Quantity	(CEN_OF
Revenue	(CEN_OF
Cost of,Goods Sold	(CEN_OF
Returns	(CEN_OF

Figure 60: Filter on date

Note: Though it is not required for this document, you can build an SQL calculation using any ANSI SQL function. Simply click the *Functions* tab to see the available functions and arguments.

Click OK to complete the filtering expression.

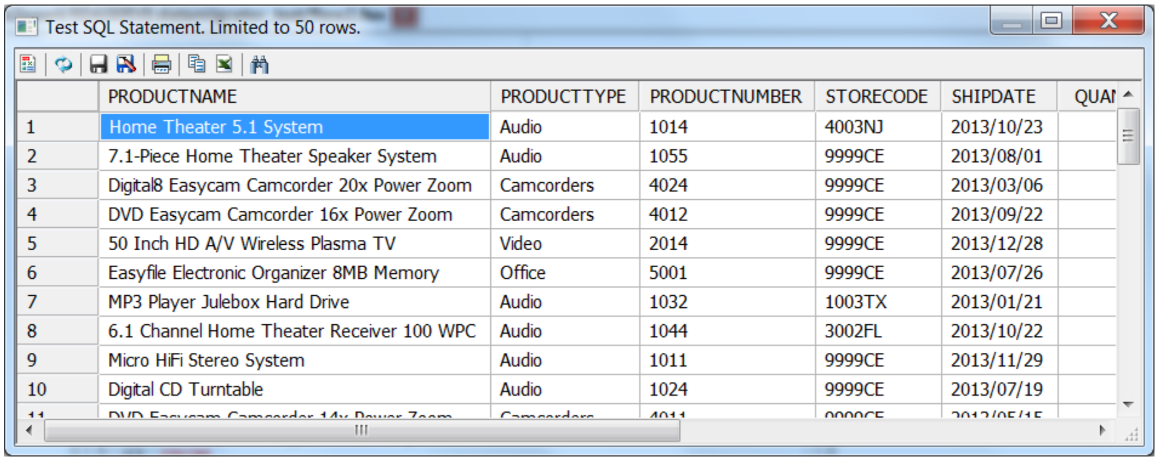
Right-click the *SQL* object and click *SQL Statement* to see the actual SQL that will execute.



```
SELECT
  T1.PRODUCTNAME ,
  T1.PRODUCTTYPE ,
  T1.PRODUCTNUMBER ,
  T2.STORECODE ,
  T2.SHIPDATE ,
  SUM(T2.QUANTITY ) ,
  SUM(T2.LINETOTAL ) ,
  SUM(T2.COSTOFGOODSSOLD ) ,
  SUM(T2.LINETOTAL - T2.COSTOFGOODSSOLD )  AS PROFIT ,
  AVG(T2.SHIP_DIFFERENCE )
FROM
  (cen_inventory T1  INNER JOIN cen_orders T2
     ON
    T1.PRODUCTNUMBER = T2.PRODUCTNUMBER )
 WHERE
  T2.SHIPDATE >= '2013-01-01'
 GROUP BY
  T1.PRODUCTNAME ,
  T1.PRODUCTTYPE ,
  T1.PRODUCTNUMBER ,
  T2.STORECODE ,
  T2.SHIPDATE
```

Figure 61: SQL Statement

Click the *Test SQL Statement* button to see the results.



	PRODUCTNAME	PRODUCTTYPE	PRODUCTNUMBER	STORECODE	SHIPDATE	QUAI
1	Home Theater 5.1 System	Audio	1014	4003NJ	2013/10/23	
2	7.1-Piece Home Theater Speaker System	Audio	1055	9999CE	2013/08/01	
3	Digital8 Easycam Camcorder 20x Power Zoom	Camcorders	4024	9999CE	2013/03/06	
4	DVD Easycam Camcorder 16x Power Zoom	Camcorders	4012	9999CE	2013/09/22	
5	50 Inch HD A/V Wireless Plasma TV	Video	2014	9999CE	2013/12/28	
6	Easyfile Electronic Organizer 8MB Memory	Office	5001	9999CE	2013/07/26	
7	MP3 Player Julebox Hard Drive	Audio	1032	1003TX	2013/01/21	
8	6.1 Channel Home Theater Receiver 100 WPC	Audio	1044	3002FL	2013/10/22	
9	Micro HiFi Stereo System	Audio	1011	9999CE	2013/11/29	
10	Digital CD Turntable	Audio	1024	9999CE	2013/07/19	

Figure 62: Test SQL Statement output

> **Note:** The number of records retrieved will depend on the Run Options set from *Tools* and *Options*.

Click the red X to close the Test SQL Statement window, and click OK to close the *Select Statement* window.

Click the *Save* button to save our data flow up to this point.

Adding Data Targets

We are now ready to create the data targets into which the source data will be copied, based on the mapping and rules defined in the *SQL* columns object.

First, let's get rid of the data target brought over when we copied the flow. Highlight the data target and press delete. Click *Yes* on the confirmation screen.

> **Note:** A very handy feature of the editor is that you can undo changes. If you inadvertently delete something you wanted to keep, click the *Undo* button on the top left of the DMC window or hit the *ctrl-z* key sequence to undo back to what was there before.

At the beginning of this chapter we create a data target table. We will now utilize it.

We are going to add two data targets and specify the options to use when loading data into them.

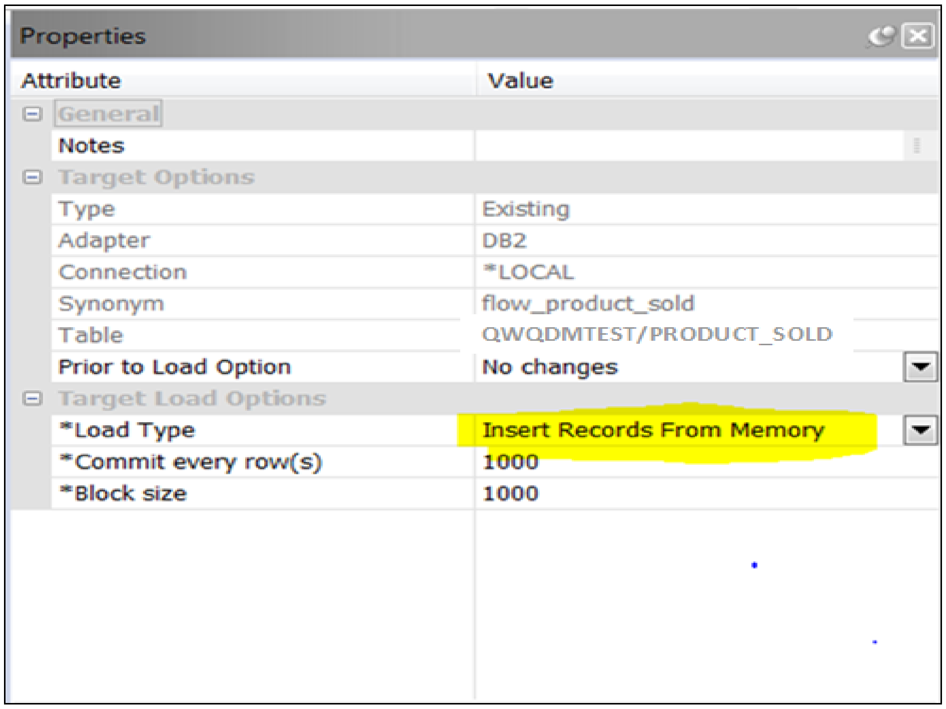
- The first data target is a pre-existing table that we will be adding data into. This represents a common situation where a data flow will be adding data into an existing data warehouse or data mart.
- The second is a target table that we will create as part of the data flow. It represents a transient table used for a fixed period of time and then ultimately discarded.

To specify the first data target:

1. From the navigation pane, drag the synonym **flow_product_sold** into the workspace, to the right of the SQL object. (The position to the right of the SQL object makes it a data target).
2. Once you have added the data target to the data flow, you can specify how incoming data should be handled during the loading process.

Right-click the **flow_product_sold** target object and click *Properties*. For most of the properties we can stay with the defaults. However, for *Load Type* specify *Insert Records From Memory*.

> **Note**: Using Load Type of *Insert Records From Memory* is a good habit to get into for DataMigrator for i, as it generally has the best performance characteristics for 'bulk' load scenarios such as this.



Properties

Attribute	Value
General	
Notes	
Target Options	
Type	Existing
Adapter	DB2
Connection	*LOCAL
Synonym	flow_product_sold
Table	QWQDMTEST/PRODUCT_SOLD
Prior to Load Option	No changes
Target Load Options	
*Load Type	Insert Records From Memory
*Commit every row(s)	1000
*Block size	1000

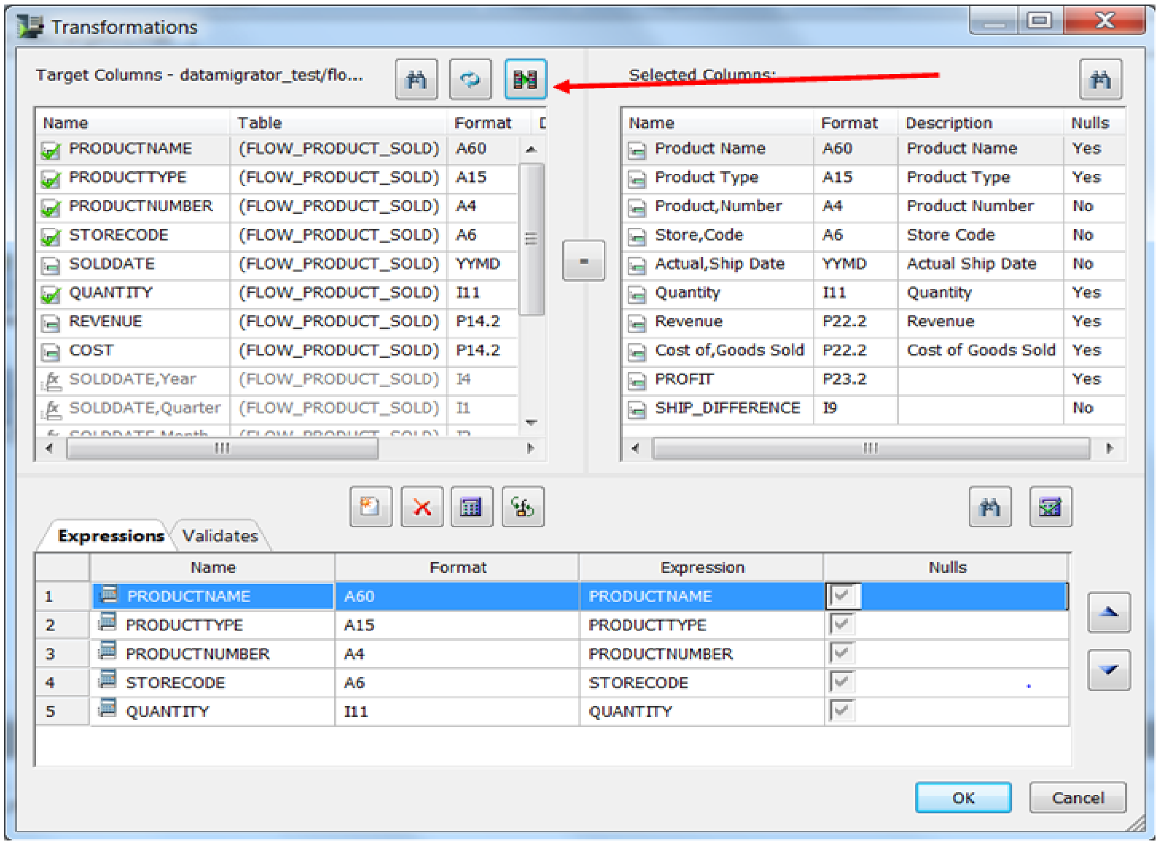
Figure 63: Existing data target properties

> **Note:** If you want to clear the target table before each run, the properties can be used to control that. In the *Prior to Load Option* on the *Properties* window, choose the option *Delete all rows from table*.

Click **X** to close the Properties window.

The next step is to map the data source columns to the data target columns.

1. Right-click **flow_product_sold** and this time, click *Target Transformations*. The Transformations window opens.
2. Click the *Automap* button.



Target Columns - datamigrator_test/flo...

Name	Table	Format
PRODUCTNAME	(FLOW_PRODUCT_SOLD)	A60
PRODUCTTYPE	(FLOW_PRODUCT_SOLD)	A15
PRODUCTNUMBER	(FLOW_PRODUCT_SOLD)	A4
STORECODE	(FLOW_PRODUCT_SOLD)	A6
SOLDDATE	(FLOW_PRODUCT_SOLD)	YYMD
QUANTITY	(FLOW_PRODUCT_SOLD)	I11
REVENUE	(FLOW_PRODUCT_SOLD)	P14.2
COST	(FLOW_PRODUCT_SOLD)	P14.2
SOLDDATE,Year	(FLOW_PRODUCT_SOLD)	I4
SOLDDATE,Quarter	(FLOW_PRODUCT_SOLD)	I1

Selected Columns:

Name	Format	Description	Nulls
Product Name	A60	Product Name	Yes
Product Type	A15	Product Type	Yes
Product,Number	A4	Product Number	No
Store,Code	A6	Store Code	No
Actual,Ship Date	YYMD	Actual Ship Date	No
Quantity	I11	Quantity	Yes
Revenue	P22.2	Revenue	Yes
Cost of,Goods Sold	P22.2	Cost of Goods Sold	Yes
PROFIT	P23.2		Yes
SHIP_DIFFERENCE	I9		No

	Name	Format	Expression	Nulls
1	PRODUCTNAME	A60	PRODUCTNAME	✓
2	PRODUCTTYPE	A15	PRODUCTTYPE	✓
3	PRODUCTNUMBER	A4	PRODUCTNUMBER	✓
4	STORECODE	A6	STORECODE	✓
5	QUANTITY	I11	QUANTITY	✓

Figure 64: Target transformation using automap button

The five columns with identical names and data types are mapped and moved to the *Expressions* tab. Note that the mapped columns are also selected in green in the *Target Columns* list.

Three target column names are deselected, so you need to create mappings or transformations for them. In the *Target Columns* list, **SOLDDATE** contains sold date. You will map it to the **Actual Ship Date** column in the Selected Columns list.

1. Under *Target Columns*, click **SOLDDATE**.
2. Under *Selected Columns*, click **Actual Ship Date**.
3. Click the equal sign (=) to move the mapping into the grid in the *Expressions* tab.

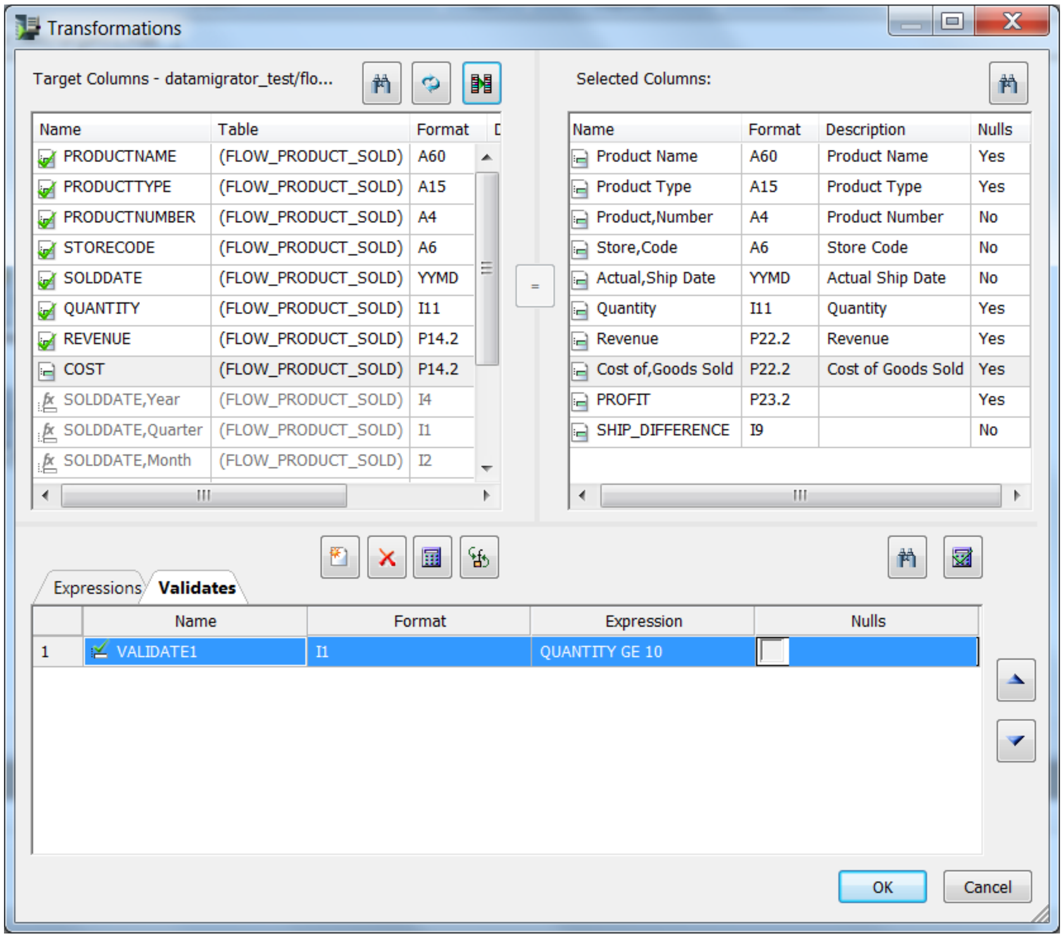
Repeat these steps for **REVENUE** to **Revenue** and **COST** to **Cost of Goods Sold**.

> **Note:** If we needed to do any calculations or expressions at this point we would use the *Insert Transformations* button to create the transformation(s), similar to what we did when we created the **PROFIT** source transformation earlier. If we did the calculation here it would be called a *Target Transformation*.

Adding Validation

At this point we will add some validation to the incoming data. To do that, we want to load only those records with a quantity greater than or equal to ten. Records that do not meet this validation criterion can be logged to a file, for additional processing, or for review at a later date.

Click the *Validates* tab in the Transformation window. Then click the *Insert Transforms* button to open the *Transformation Calculator*. To build the validation expression, double-click **QUANTITY** in the tree, and then select *GE* from the Relation drop-down menu. Input **10** into the Value field.



Name	Table	Format
PRODUCTNAME	(FLOW_PRODUCT_SOLD)	A60
PRODUCTTYPE	(FLOW_PRODUCT_SOLD)	A15
PRODUCTNUMBER	(FLOW_PRODUCT_SOLD)	A4
STORECODE	(FLOW_PRODUCT_SOLD)	A6
SOLDDATE	(FLOW_PRODUCT_SOLD)	YYMD
QUANTITY	(FLOW_PRODUCT_SOLD)	I11
REVENUE	(FLOW_PRODUCT_SOLD)	P14.2
COST	(FLOW_PRODUCT_SOLD)	P14.2
SOLDDATE,Year	(FLOW_PRODUCT_SOLD)	I4
SOLDDATE,Quarter	(FLOW_PRODUCT_SOLD)	I1
SOLDDATE,Month	(FLOW_PRODUCT_SOLD)	I2

Name	Format	Description	Nulls
Product Name	A60	Product Name	Yes
Product Type	A15	Product Type	Yes
Product,Number	A4	Product Number	No
Store,Code	A6	Store Code	No
Actual,Ship Date	YYMD	Actual Ship Date	No
Quantity	I11	Quantity	Yes
Revenue	P22.2	Revenue	Yes
Cost of,Goods Sold	P22.2	Cost of Goods Sold	Yes
PROFIT	P23.2		Yes
SHIP_DIFFERENCE	I9		No

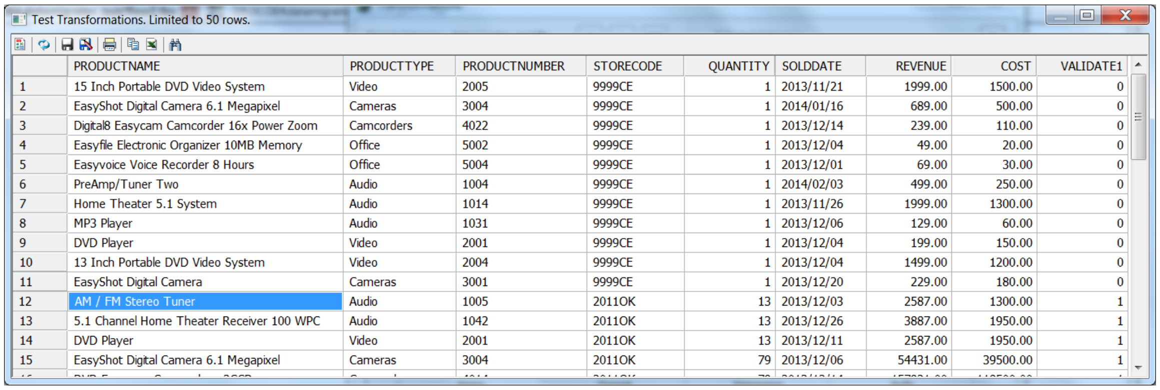
	Name	Format	Expression	Nulls
1	VALIDATE1	I1	QUANTITY GE 10	☐

Figure 65: Adding validation

Click OK.

Let's test the validation and transformations to ensure that they are syntactically correct and performing the desired calculations. The test retrieves some rows from the server, applies the transformations, and displays the results.

Click the *Test Transforms* button in the upper-right corner, above the *Validates* tab.


Test Transformations. Limited to 50 rows.

	PRODUCTNAME	PRODUCTTYPE	PRODUCTNUMBER	STORECODE	QUANTITY	SOLDDATE	REVENUE	COST	VALIDATE1
1	15 Inch Portable DVD Video System	Video	2005	9999CE	1	2013/11/21	1999.00	1500.00	0
2	EasyShot Digital Camera 6.1 Megapixel	Cameras	3004	9999CE	1	2014/01/16	689.00	500.00	0
3	Digital8 Easycam Camcorder 16x Power Zoom	Camcorders	4022	9999CE	1	2013/12/14	239.00	110.00	0
4	Easyfile Electronic Organizer 10MB Memory	Office	5002	9999CE	1	2013/12/04	49.00	20.00	0
5	Easyvoice Voice Recorder 8 Hours	Office	5004	9999CE	1	2013/12/01	69.00	30.00	0
6	PreAmp/Tuner Two	Audio	1004	9999CE	1	2014/02/03	499.00	250.00	0
7	Home Theater 5.1 System	Audio	1014	9999CE	1	2013/11/26	1999.00	1300.00	0
8	MP3 Player	Audio	1031	9999CE	1	2013/12/06	129.00	60.00	0
9	DVD Player	Video	2001	9999CE	1	2013/12/04	199.00	150.00	0
10	13 Inch Portable DVD Video System	Video	2004	9999CE	1	2013/12/04	1499.00	1200.00	0
11	EasyShot Digital Camera	Cameras	3001	9999CE	1	2013/12/20	229.00	180.00	0
12	AM / FM Stereo Tuner	Audio	1005	2011OK	13	2013/12/03	2587.00	1300.00	1
13	5.1 Channel Home Theater Receiver 100 WPC	Audio	1042	2011OK	13	2013/12/26	3887.00	1950.00	1
14	DVD Player	Video	2001	2011OK	13	2013/12/11	2587.00	1950.00	1
15	EasyShot Digital Camera 6.1 Megapixel	Cameras	3004	2011OK	79	2013/12/06	54431.00	39500.00	1

Figure 66: Test transformations

In the **VALIDATE1** column, the number 1 represents rows that will be accepted based on the validation test. The 0s represent rows that will be rejected because **QUANTITY** is less than 10.

> **Tip:** Since you have been working in the *Transformations* window, testing from there is the simplest method, but you can also test the transformation by right-clicking the Target object, selecting *Toggle*, and clicking the *Test Transforms* tab.

Close the Test Transformation window, and click **OK** to close the Transformations window.

Click the *Save* button to save our data flow up to this point.

Adding a Second Data Target

DataMigrator can load multiple data targets in a single data flow. Add another target object into the data flow. This time, the data target does not exist, so we will create it using the columns in the *SQL Select* statement that were defined for the *SQL Select Columns* object.

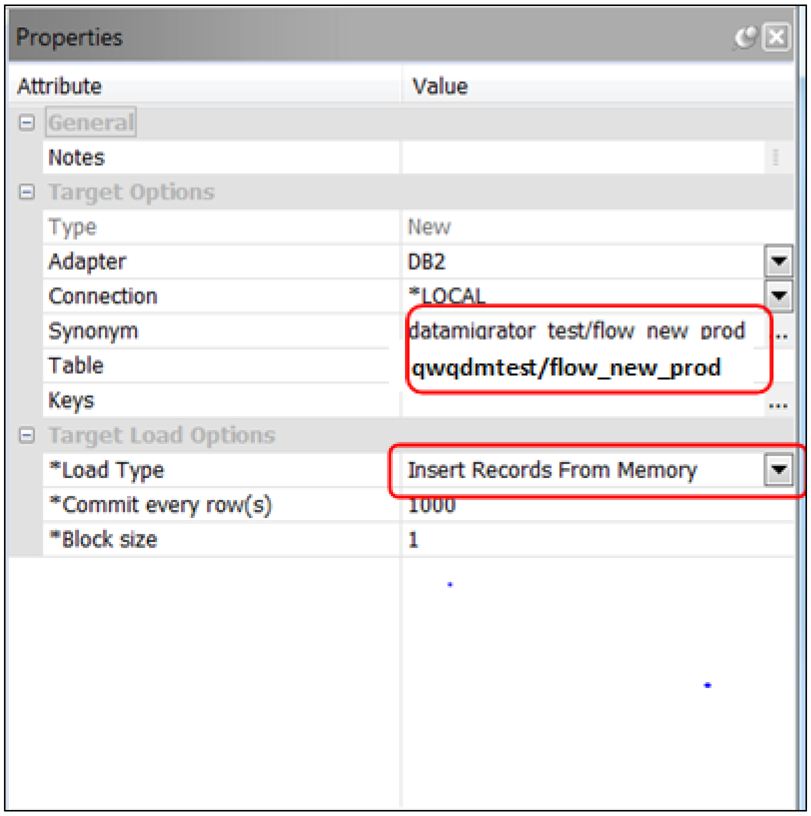
> **Note:** This flow will create the base table, but other flows can update it with additional information.

Right-click in the workspace to the right of the SQL object, select *Add target*, and then click *New*. A new data target appears to the right of the SQL object.

Right-click the new target and click *Properties*. The Target Properties window opens.

Some of the properties are prefilled with the defaults we want. The Adapter is DB2 (for DB2 for i). The Connection is *LOCAL, which means the resulting database table will be put back on the same system where DataMigrator for i is running.

For synonym specify **datamigrator_test/flow_new_prod**, which will give the name of the synonym flow_new_prod and place it in folder **datamigrator_test**. For the table specify **qwqdmtest/flow_new_prod** to make sure the table, named flow_new_prod, goes into library qwqdmtest. Finally, for Load Type specify *Insert Records From Memory*.

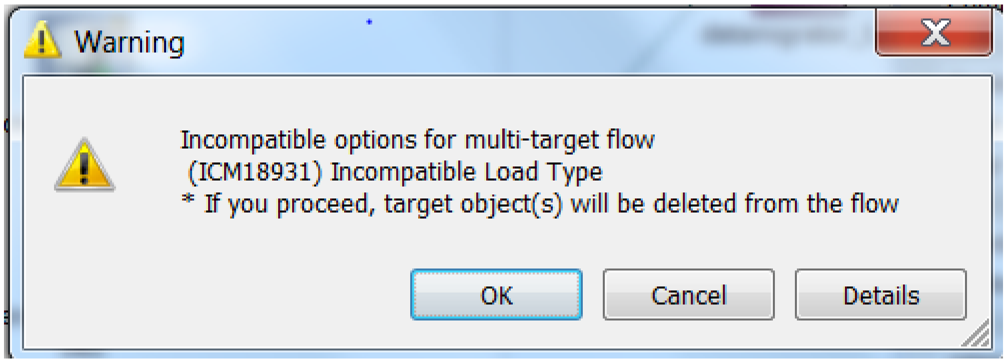


Attribute	Value
General	
Notes	
Target Options	
Type	New
Adapter	DB2
Connection	*LOCAL
Synonym	datamigrator_test/flow_new_prod
Table	qwqdmtest/flow_new_prod
Keys	
Target Load Options	
*Load Type	Insert Records From Memory
*Commit every row(s)	1000
*Block size	1

Figure 67: New data target

The properties are now set the way we want.

> **Note**: The properties options identified with an asterisk (*) are flow wide properties. That means they have to be set the same for all data targets. Since we have two data targets in this flow, we have to make sure these asterisk options are the same. If the options are not consistent you will get a warning when you attempt to save or run the flow:



Warning

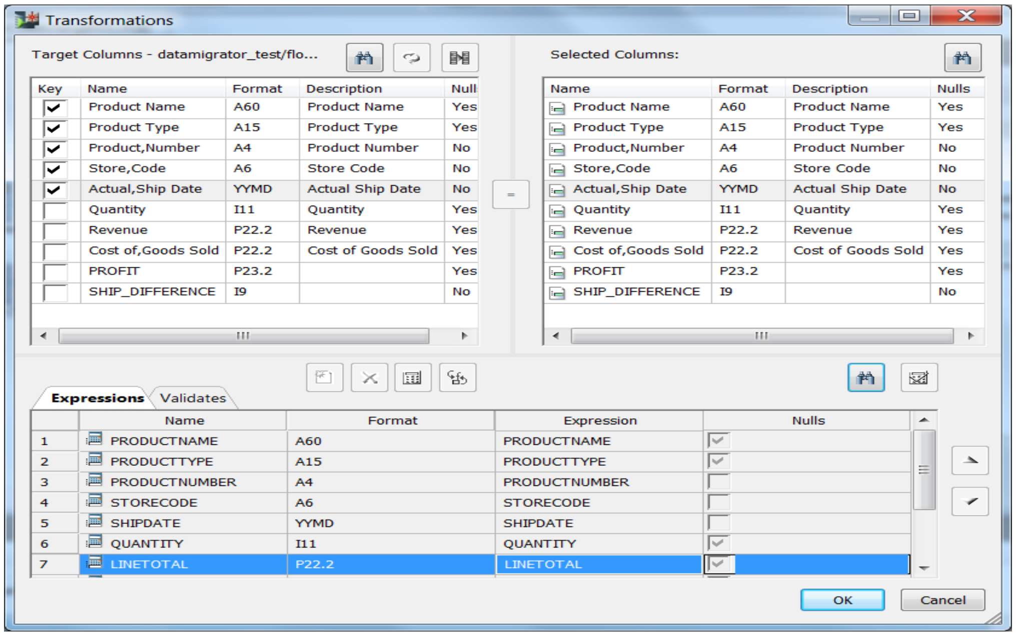
Incompatible options for multi-target flow
(ICM18931) Incompatible Load Type
* If you proceed, target object(s) will be deleted from the flow

Figure 68: Warning when flow wide attributes are not in sync

Close the properties by clicking the X on the properties window.

Now, right-click the new data target and click *Target Transformations*.

In the table you are creating, PRODUCTNAME, PRODUCTTYPE, PRODUCTNUMBER, STORECODE and SHIPDATE are all keys. This means that each resulting row has a unique combination of those columns. Identifying this allows DataMigrator for i to add database enforcement (unique index) to the underlying table and to utilize that unique aspect in subsequent change data processing. While not ultimately required for the data flow, identification of the keys of the target helps ensure data integrity. Click each of these columns to identify it as part of the key.



Transformations

Target Columns - datamigrator_test/flo...

Key	Name	Format	Description	Null
✓	Product Name	A60	Product Name	Yes
✓	Product Type	A15	Product Type	Yes
✓	Product,Number	A4	Product Number	No
✓	Store,Code	A6	Store Code	No
✓	Actual,Ship Date	YYMD	Actual Ship Date	No
	Quantity	I11	Quantity	Yes
	Revenue	P22.2	Revenue	Yes
	Cost of,Goods Sold	P22.2	Cost of Goods Sold	Yes
	PROFIT	P23.2		Yes
	SHIP_DIFFERENCE	I9		No

Selected Columns:

Name	Format	Description	Nulls
Product Name	A60	Product Name	Yes
Product Type	A15	Product Type	Yes
Product,Number	A4	Product Number	No
Store,Code	A6	Store Code	No
Actual,Ship Date	YYMD	Actual Ship Date	No
Quantity	I11	Quantity	Yes
Revenue	P22.2	Revenue	Yes
Cost of,Goods Sold	P22.2	Cost of Goods Sold	Yes
PROFIT	P23.2		Yes
SHIP_DIFFERENCE	I9		No

Expressions | Validates

	Name	Format	Expression	Nulls
1	PRODUCTNAME	A60	PRODUCTNAME	✓
2	PRODUCTTYPE	A15	PRODUCTTYPE	✓
3	PRODUCTNUMBER	A4	PRODUCTNUMBER	
4	STORECODE	A6	STORECODE	
5	SHIPDATE	YYMD	SHIPDATE	
6	QUANTITY	I11	QUANTITY	✓
7	LINETOTAL	P22.2	LINETOTAL	✓

Figure 69: Specifying keys on target transformations

Click OK to return to the *Data Flow* tab, which now contains two data sources and two data targets.

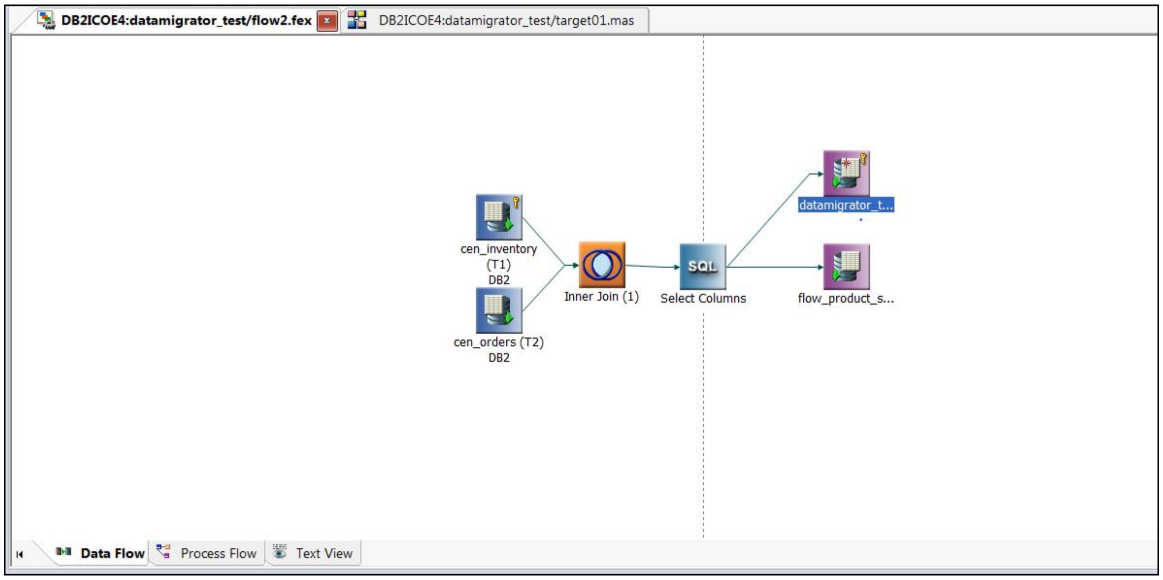


Figure 70: Completed data flow with two data targets

Click the *Save* button to save our data flow.

Running the Flow

Our data flow is now ready. Both data targets **flow_new_prod** and **flow_product_sold** will be loaded with data based on the data flow specifications when the data flow is run or submitted.

Let’s run the flow.

- In the left navigation tree, find the flow **flow2** under the datamigrator_test folder. Right-click on flow2 and select *Submit* from the menu.

  or

- In the ribbon, click *Run* and select *Submit* from the drop-down menu.

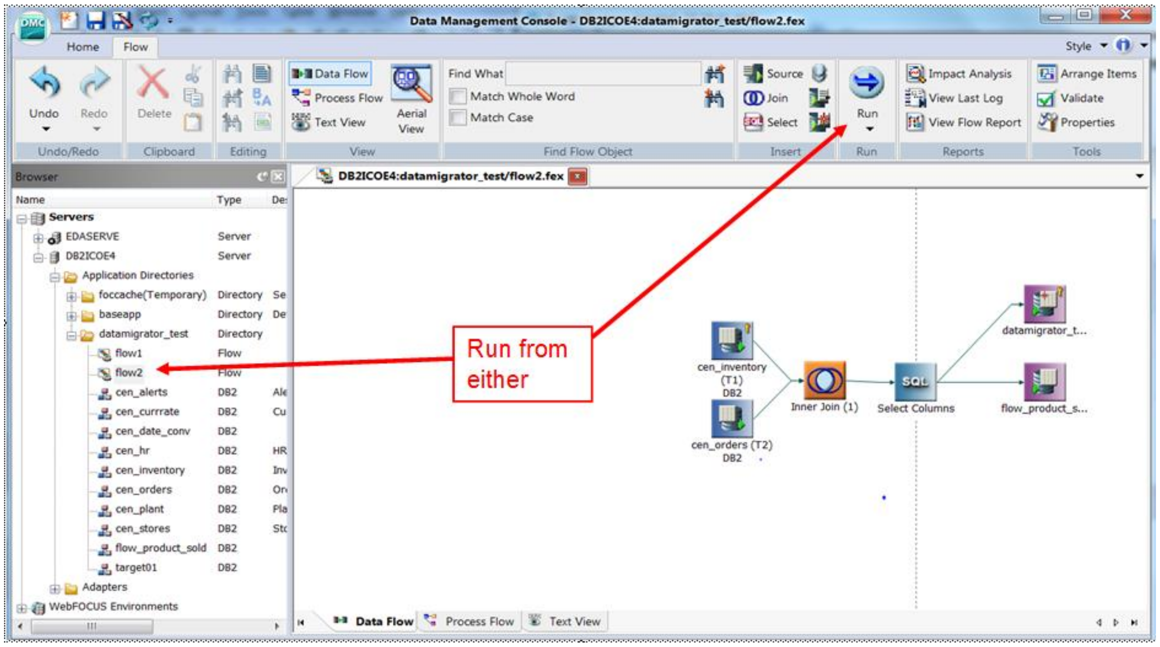
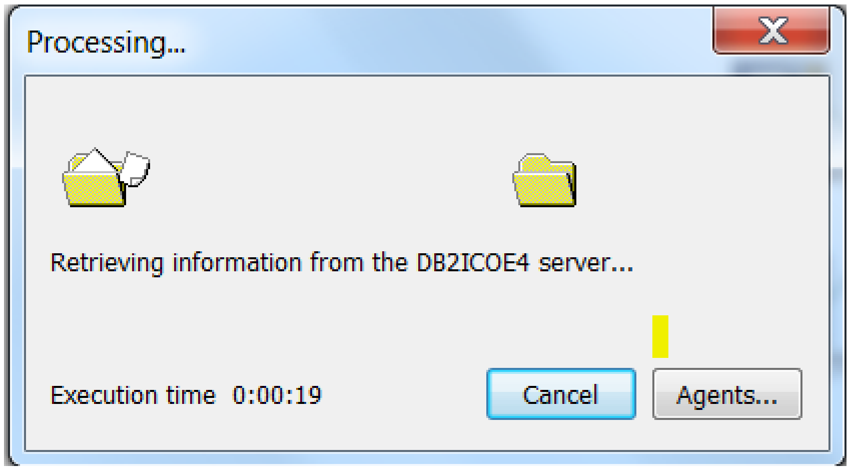


Figure 71: Submit flow from one of two places

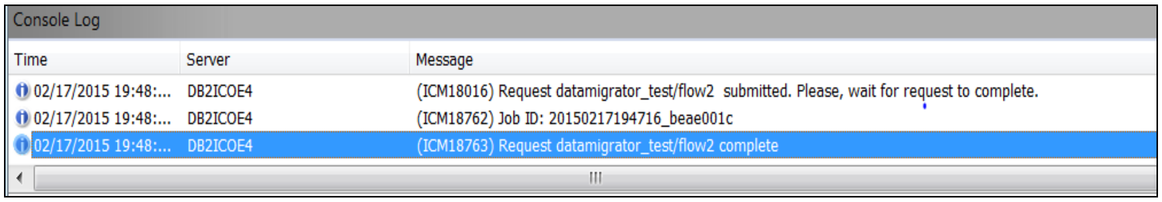


Retrieving information from the DB2ICOE4 server...

Execution time 0:00:19

Figure 72: Flow execution in progress.

When the execution completes, you can review the console log at the bottom of the DMC window.



Time	Server	Message
02/17/2015 19:48:...	DB2ICOE4	(ICM18016) Request datamigrator_test/flow2 submitted. Please, wait for request to complete.
02/17/2015 19:48:...	DB2ICOE4	(ICM18762) Job ID: 20150217194716_beae001c
02/17/2015 19:48:...	DB2ICOE4	(ICM18763) Request datamigrator_test/flow2 complete

Figure 73: Console Log

When a flow is executed with a **Submit** rather than **Run**, the flow is run in a background process. Consequently the console log shows only that the request was submitted and that it was successful. To get more details on the process itself, we need to look at the run log from the process. The log is also important for analyzing any problems that may have occurred in the flow run.

> **Note:** In general you should perform a Run of a flow when you are building and testing it. This will cause the flow to run in your process and all log details will show up in the Console log. Once a flow is finished, a Submit can be used to have a background job run it, thereby freeing the DMC session for other work.

To view the run log, click on the *View Last Log* option in the ribbon.

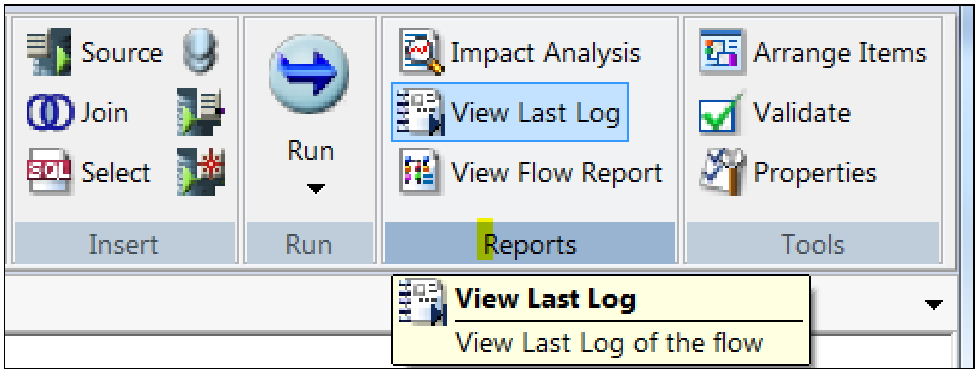
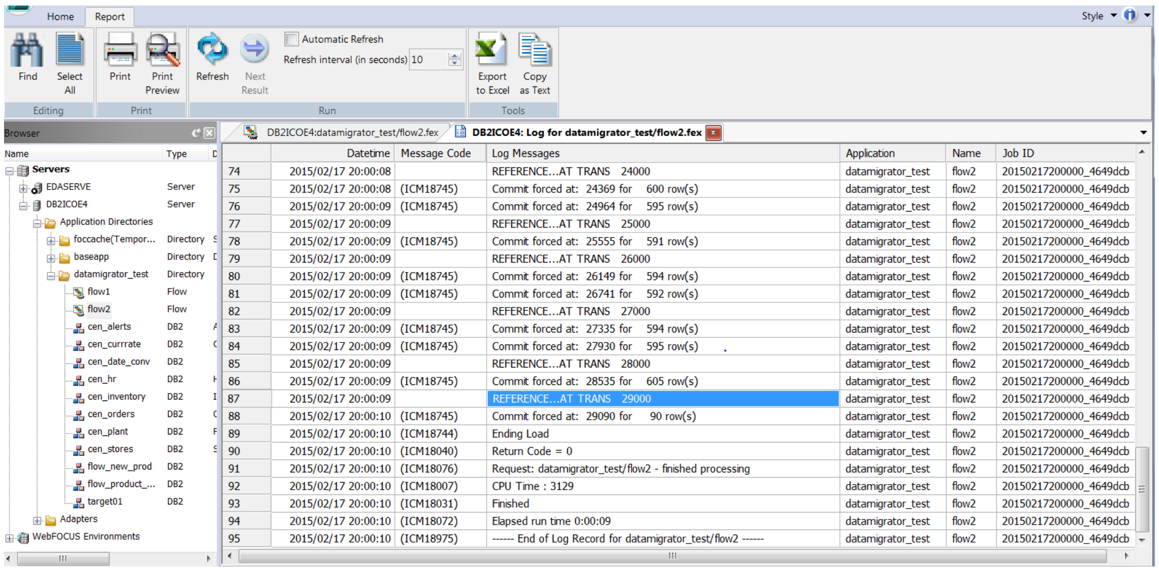


Figure 74: View last log option



	Datetime	Message Code	Log Messages	Application	Name	Job ID
74	2015/02/17 20:00:08		REFERENCE...AT TRANS 24000	datamigrator_test	flow2	20150217200000_4649dcb
75	2015/02/17 20:00:08	(ICM18745)	Commit forced at: 24369 for 600 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
76	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 24964 for 595 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
77	2015/02/17 20:00:09		REFERENCE...AT TRANS 25000	datamigrator_test	flow2	20150217200000_4649dcb
78	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 25555 for 591 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
79	2015/02/17 20:00:09		REFERENCE...AT TRANS 26000	datamigrator_test	flow2	20150217200000_4649dcb
80	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 26149 for 594 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
81	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 26741 for 592 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
82	2015/02/17 20:00:09		REFERENCE...AT TRANS 27000	datamigrator_test	flow2	20150217200000_4649dcb
83	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 27335 for 594 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
84	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 27930 for 595 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
85	2015/02/17 20:00:09		REFERENCE...AT TRANS 28000	datamigrator_test	flow2	20150217200000_4649dcb
86	2015/02/17 20:00:09	(ICM18745)	Commit forced at: 28535 for 605 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
87	2015/02/17 20:00:09		REFERENCE...AT TRANS 29000	datamigrator_test	flow2	20150217200000_4649dcb
88	2015/02/17 20:00:10	(ICM18745)	Commit forced at: 29090 for 90 row(s)	datamigrator_test	flow2	20150217200000_4649dcb
89	2015/02/17 20:00:10	(ICM18744)	Ending Load	datamigrator_test	flow2	20150217200000_4649dcb
90	2015/02/17 20:00:10	(ICM18040)	Return Code = 0	datamigrator_test	flow2	20150217200000_4649dcb
91	2015/02/17 20:00:10	(ICM18076)	Request: datamigrator_test/flow2 - finished processing	datamigrator_test	flow2	20150217200000_4649dcb
92	2015/02/17 20:00:10	(ICM18007)	CPU Time : 3129	datamigrator_test	flow2	20150217200000_4649dcb
93	2015/02/17 20:00:10	(ICM18031)	Finished	datamigrator_test	flow2	20150217200000_4649dcb
94	2015/02/17 20:00:10	(ICM18072)	Elapsed run time 0:00:09	datamigrator_test	flow2	20150217200000_4649dcb
95	2015/02/17 20:00:10	(ICM18975)	------ End of Log Record for datamigrator_test/flow2 ------	datamigrator_test	flow2	20150217200000_4649dcb

Figure 75: First page of the run log

Your log's contents may not look exactly the same as the log shown here. There are environmental factors that affect what is produced in the log. The important thing is that a *Return Code = 0* entry and a *Finished* entry appear to indicate success.

> **Note**: Notice that the ribbon options have changed to be specific to the log. This is another example of how the ribbon changes based on context.

Chapter 5: Creating Process Flows

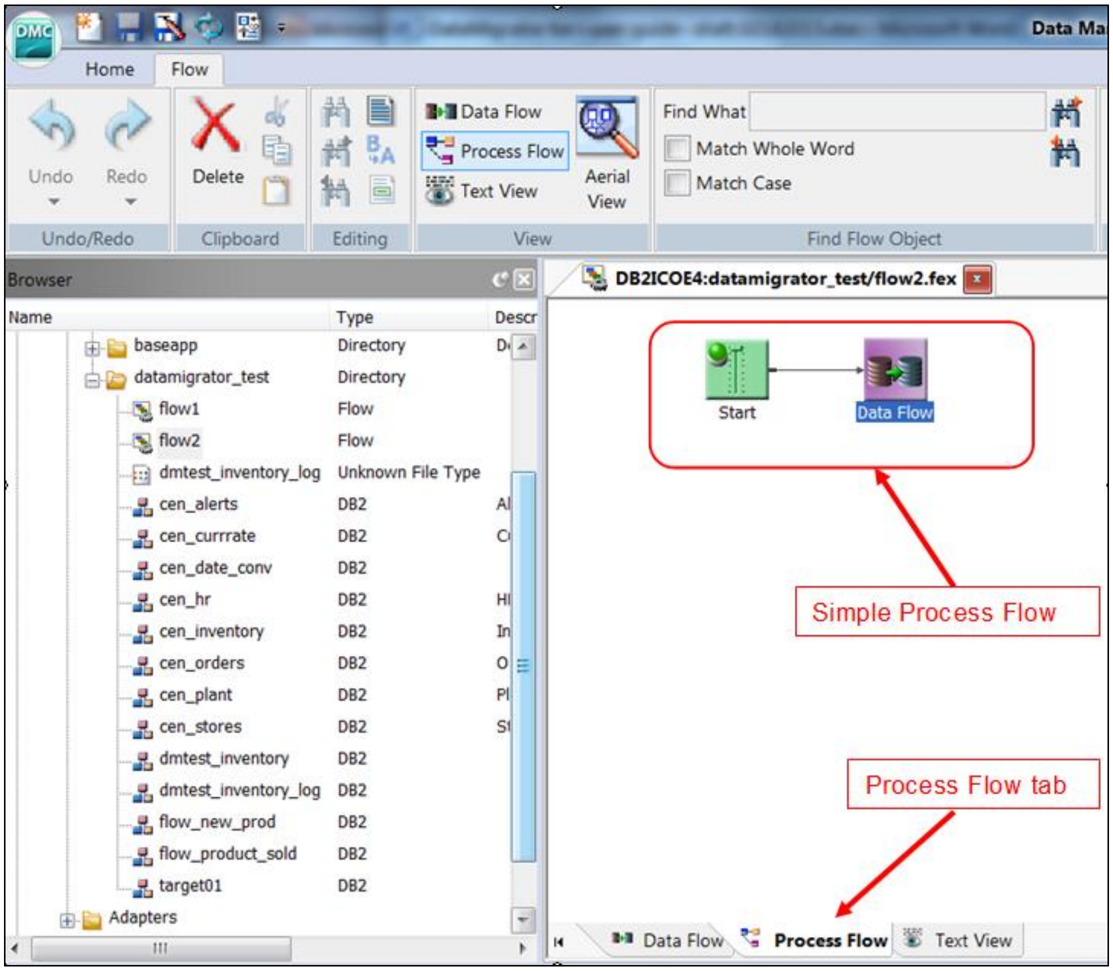
Overview

A Process Flow controls how one or more data flows will be processed. The order of the process is defined by positioning a set of objects in the workspace and defining their interactions. A process flow contains:

- A *Start* object that defines where to beging in the process flow.
- *Data Flow* objects that indicate what data to extract, and copy from data sources to data targets.
- *E-mail* objects that notify users about the status of the process at specified points.
- *Stored Procedure* objects that perform a variety of supplementary tasks before or after the extraction, and load steps defined in a data flow.
    - **Note**: these are DataMigrator for i stored procedures, not database stored procedures.
- *Connector* objects, represented as arrows, that specify execution logic for the other objects included in the process flow. For example, it indicates what to do next when a step in the process succeeds or fails.
- *Group* objects that specify the flow of a subset of objects within the total flow. For example, two objects in a group that should be processed simultaneously, rather than sequentially, with the object that follows next. This could, for example, be an email notification that is waiting until processing has been completed for both of the grouped objects.

As mentioned before, every data flow automatically has a process flow created for it. When you run or submit a data flow, it is the process flow itself that is being executed. The process flow, in turn, drives the data flow.

We can see this with the flow2 dataflow created in the last chapter. There is a tab at the bottom of the workspace for the data flow that says *Process Flow*. Click on that to see the process flow that was automatically generated for the data flow.



Name	Type
baseapp	Directory
datamigrator_test	Directory
flow1	Flow
flow2	Flow
dmtest_inventory_log	Unknown File Type
cen_alerts	DB2
cen_currrate	DB2
cen_date_conv	DB2
cen_hr	DB2
cen_inventory	DB2
cen_orders	DB2
cen_plant	DB2
cen_stores	DB2
dmtest_inventory	DB2
dmtest_inventory_log	DB2
flow_new_prod	DB2
flow_product_sold	DB2
target01	DB2
Adapters	

Figure 76: Process flow of a data flow

In the simplest case, you don’t need to create a process flow at all, DataMigrator for i does it for you. However, there are many occasions where you will want to add more steps in the flow: combining multiple data flows into one process, generate email feedback, handle errors, etc…

Creating a Process Flow

For this exercise, we will be creating a process flow that uses the data flow created in the last exercise, and then branches to one of two procedures based on the success or failure of the execution of that data flow. The appropriate results will be recorded in a log to facilitate quick trouble-shooting, if necessary, or immediate communication to indicate success.

> **Note:** Although not demonstrated in this exercise, you can extend the success and failure branches of process flow to trigger the distribution of email messages, either to those in charge of correcting errors (upon failure), or to those who need the current data (upon success).

The simplest way to proceed would be to continue working with flow2's process flow. We could add other elements to enhance the process flow. In some situations, that might in fact be sufficient.

However, this is not necessarily the best way to proceed. Instead, we will use a method that is more *modular* and *flexible*. We will start a new flow, then click the *Process Flow* tab and drag the Data Flow object into the Process Flow workspace in the correct position in the flow. With this method, each flow is saved separately; a Data Flow object can then be used in more than one Process Flow, and a Process Flow can be modified as needed and used to incorporate different data flow objects.

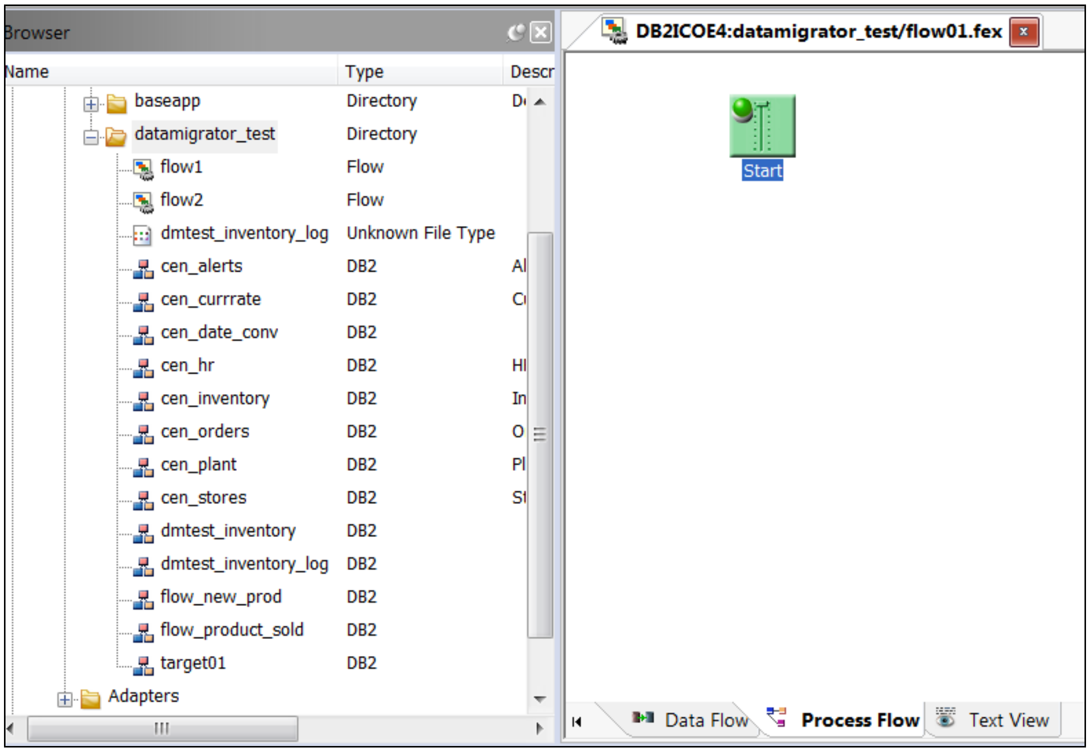
To create a process flow, we start the same way as we did for a data flow:

1. Right-click the **datamigrator_test** folder, select *New*, and then click *Flow*.

   A workspace opens in the right pane. The *Data Flow* tab is active.

2. Click the *Process Flow* tab to change design mode.

   To anchor the Process flow, the Start object is automatically added to the workspace.



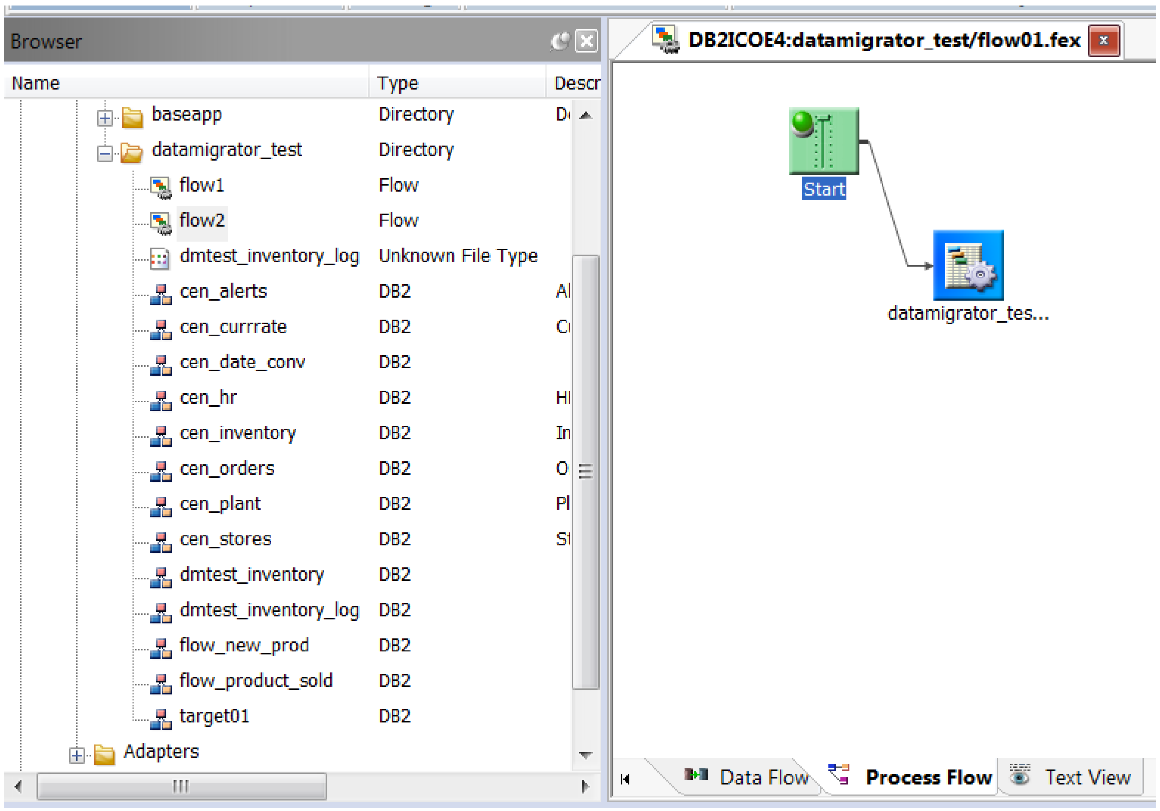
Name	Type	Descr
baseapp	Directory	D
datamigrator_test	Directory	
flow1	Flow	
flow2	Flow	
dmtest_inventory_log	Unknown File Type	
cen_alerts	DB2	Al
cen_currrate	DB2	C
cen_date_conv	DB2	
cen_hr	DB2	Hl
cen_inventory	DB2	In
cen_orders	DB2	O
cen_plant	DB2	Pl
cen_stores	DB2	St
dmtest_inventory	DB2	
dmtest_inventory_log	DB2	
flow_new_prod	DB2	
flow_product_sold	DB2	
target01	DB2	
Adapters		

Figure 77: Beginning a new process flow

Add data flow **flow2** to the process flow:

1. Drag **flow2** from the navigation tree into the Process Flow workspace to the right of the *Start* object.
2. Right-click the *Start* object in the workspace and drag toward the *Data Flow* object, forming an arrow that connects the two objects.

   Note: Move your cursor well into the flow2 icon so the arrow connection is made. Do not stop at the edge of the icon or the arrow will not connect.



Name	Type	Descr
baseapp	Directory	Di
datamigrator_test	Directory	
flow1	Flow	
flow2	Flow	
dmtest_inventory_log	Unknown File Type	
cen_alerts	DB2	Al
cen_currrate	DB2	Ci
cen_date_conv	DB2	
cen_hr	DB2	HI
cen_inventory	DB2	In
cen_orders	DB2	O
cen_plant	DB2	Pl
cen_stores	DB2	St
dmtest_inventory	DB2	
dmtest_inventory_log	DB2	
flow_new_prod	DB2	
flow_product_sold	DB2	
target01	DB2	
Adapters		

Figure 78: Drag in flow2 and connect to Start

Click the *Save* button. The *Save Procedure As* window opens. Name the Process Flow **flow2_process**. Click *Save* to return to the *Process Flow* tab.

Adding Procedures and Conditions

Next, we will create two stored procedures. We will connect these procedures to the data flow object so that one will run when the data flow executes successfully, and the other one will run when the data flow fails to execute properly.

Following the same modular model we have been using, we will create each of the stored procedures as a separate file that can be used in this process flow and others.

First, create a procedure that will appear in the log when the data flow executes correctly.

1. Right-click the **datamigrator_test** folder, select *New*, and then click *Procedure*.
2. A text editing window opens. Enter the following line of code:

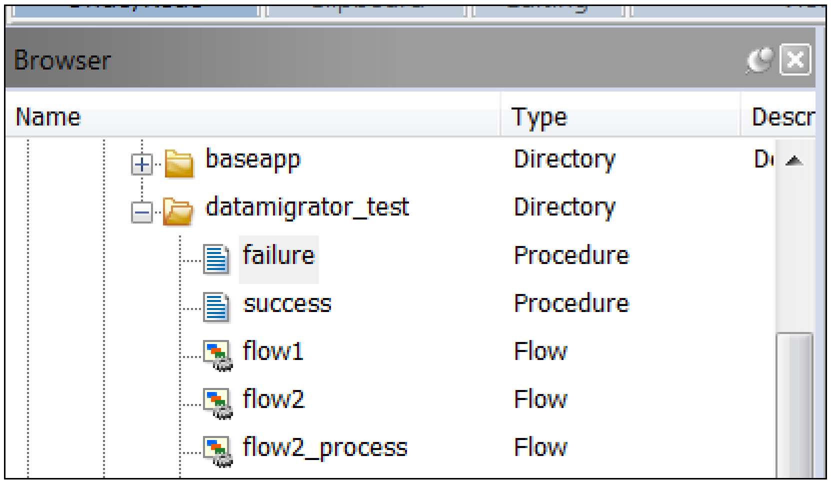
```
-TYPE SUCCESS!
```

3. Click the *Save As* button and name this procedure **success**. Click *Save* then close the editing window.
4. Right-click the **datamigrator_test** application directory again, select *New*, and then click *Procedure*.
5. The text editing window opens. Enter the code:

```
-TYPE FAILURE!
```

6. Click the *Save As* button and name this procedure **failure**. Click *Save* then close the editing window.

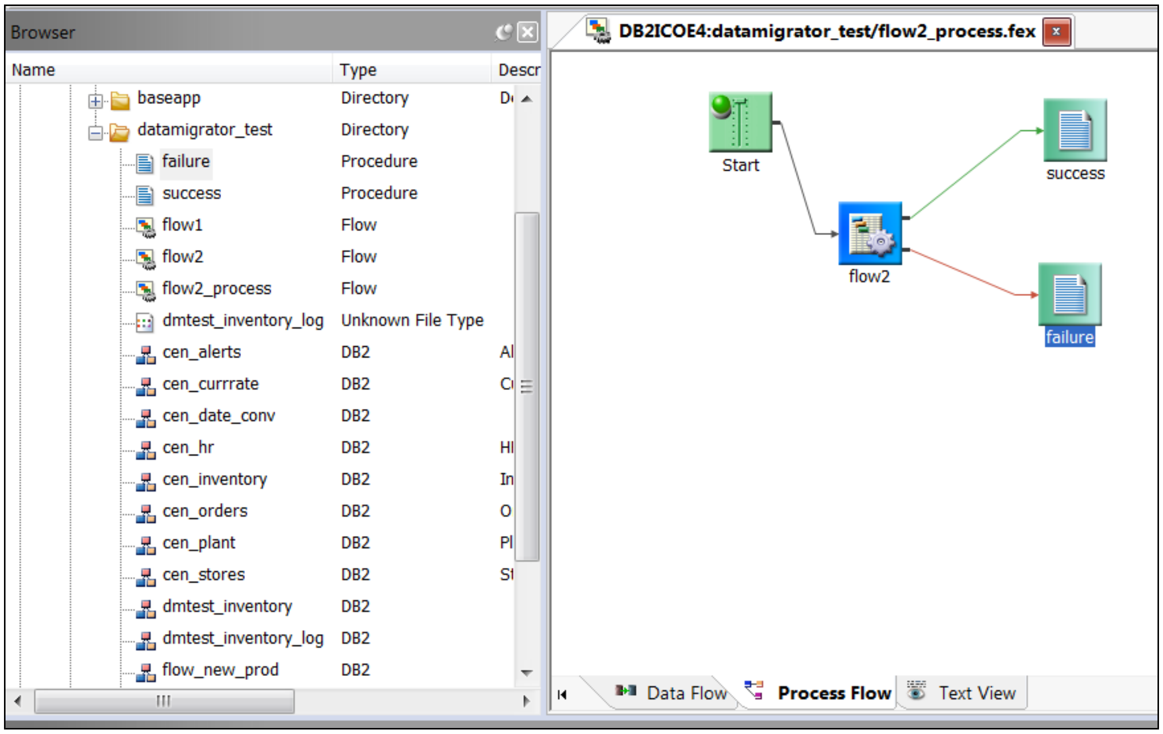
Both procedures are now listed in the **datamigrator_test** folder.



Name	Type	Descr
baseapp	Directory	D
datamigrator_test	Directory	
failure	Procedure	
success	Procedure	
flow1	Flow	
flow2	Flow	
flow2_process	Flow	

Figure 79: Success and Failure procedures

Now drag the **success** and **failure** procedures onto the workspace to the right of the flow2 icon. Connect flow2 to each procedure by right-clicking and dragging the connector line to each procedure, as we did earlier to connect Start to flow2.



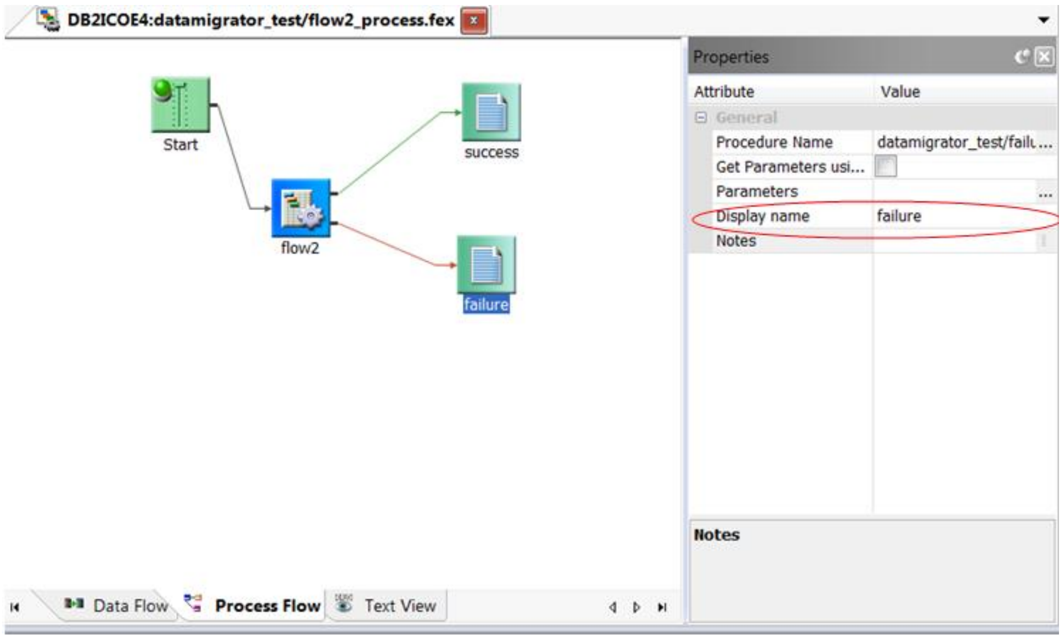
Browser

DB2ICOE4:datamigrator_test/flow2_process.fex

Name	Type	Descr
baseapp	Directory	D
datamigrator_test	Directory	
failure	Procedure	
success	Procedure	
flow1	Flow	
flow2	Flow	
flow2_process	Flow	
dmtest_inventory_log	Unknown File Type	
cen_alerts	DB2	Al
cen_currrate	DB2	C
cen_date_conv	DB2	
cen_hr	DB2	Hl
cen_inventory	DB2	In
cen_orders	DB2	O
cen_plant	DB2	Pl
cen_stores	DB2	St
dmtest_inventory	DB2	
dmtest_inventory_log	DB2	
flow_new_prod	DB2	

Figure 80: Process flow with success and failure procedures added

Note: if you notice that the labeling for the icons includes the folder name datamigrator_test, which makes it hard to identify the icons, right click the icon, select *Properties*, then change the Display name to something more readable.



Properties

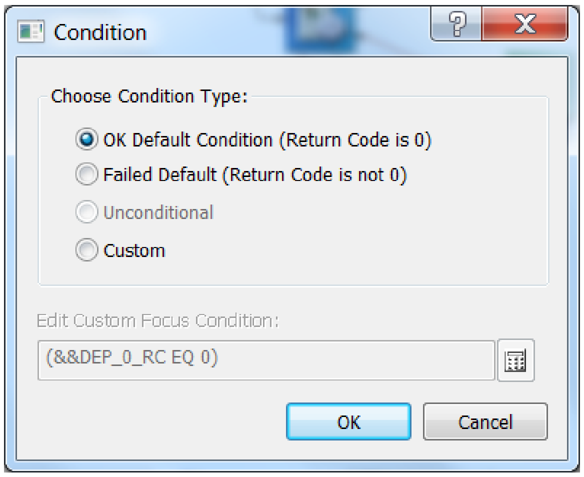
Attribute	Value
General	
Procedure Name	datamigrator_test/failu...
Get Parameters usi...	☐
Parameters	...
Display name	failure
Notes	

Notes

Figure 81: Making icon name more readable with Display name property

While we have connected the procedures, we still need to define the behavior associated with each branch of the flow. We will do that from the connector arrows.

1. Double-click the green arrow connecting the data flow object to **success**. The Condition window opens.



Condition

Choose Condition Type:

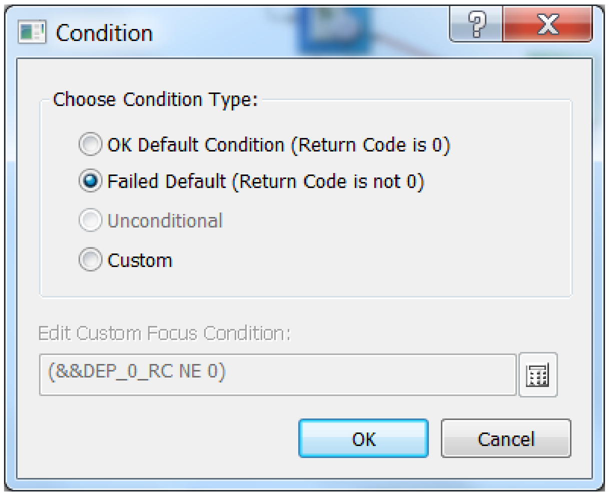
- ● OK Default Condition (Return Code is 0)
- ○ Failed Default (Return Code is not 0)
- ○ Unconditional
- ○ Custom

Edit Custom Focus Condition:

(&&DEP_0_RC EQ 0)

Figure 82: Success condition

2. *OK Default Condition (Return Code is 0)* should be selected by default.
3. Click OK.
4. Now, double-click the red arrow that connects to the failed procedure. This time, make sure *FAILED Default (Return Code is not 0)* is selected.



( ) OK Default Condition (Return Code is 0)
(●) Failed Default (Return Code is not 0)
( ) Unconditional
( ) Custom

Edit Custom Focus Condition:
(&&DEP_0_RC NE 0)

Figure 83: Fail condition

5. Click OK.

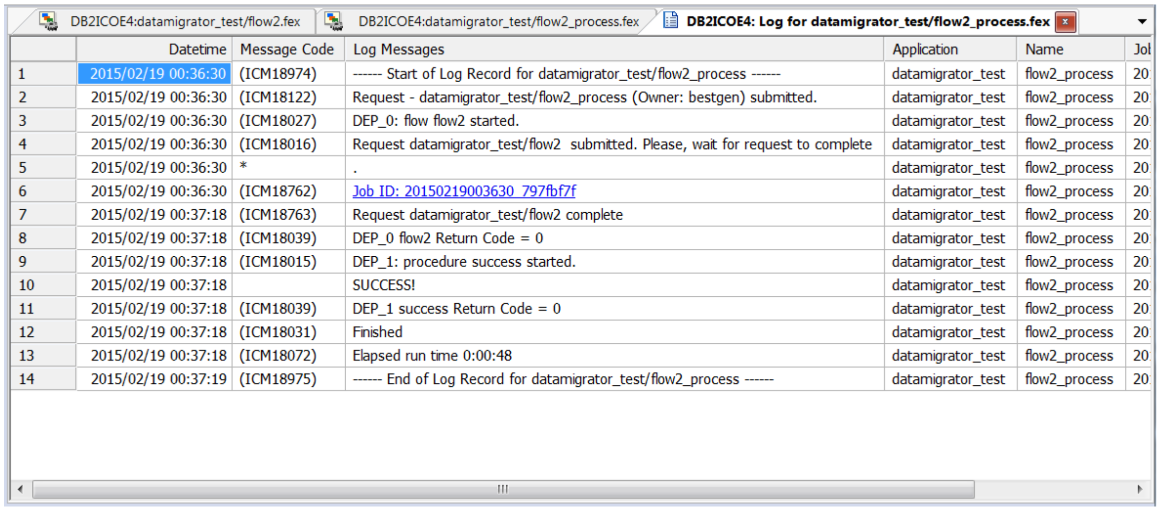
Click the *Save* button to save the process flow.

Running the Process Flow

It is time to run the flow to see if the process flow is successful, and if the proper messages are displayed.

Click *Run* from the ribbon and choose *Submit*.

Once the run completes, select *View Last Log* from the ribbon.


DB2ICOE4:datamigrator_test/flow2.fex
DB2ICOE4:datamigrator_test/flow2_process.fex
DB2ICOE4: Log for datamigrator_test/flow2_process.fex

	Datetime	Message Code	Log Messages	Application	Name	Jol
1	2015/02/19 00:36:30	(ICM18974)	------ Start of Log Record for datamigrator_test/flow2_process ------	datamigrator_test	flow2_process	20
2	2015/02/19 00:36:30	(ICM18122)	Request - datamigrator_test/flow2_process (Owner: bestgen) submitted.	datamigrator_test	flow2_process	20
3	2015/02/19 00:36:30	(ICM18027)	DEP_0: flow flow2 started.	datamigrator_test	flow2_process	20
4	2015/02/19 00:36:30	(ICM18016)	Request datamigrator_test/flow2 submitted. Please, wait for request to complete	datamigrator_test	flow2_process	20
5	2015/02/19 00:36:30	*	.	datamigrator_test	flow2_process	20
6	2015/02/19 00:36:30	(ICM18762)	Job ID: 20150219003630_797fbf7f	datamigrator_test	flow2_process	20
7	2015/02/19 00:37:18	(ICM18763)	Request datamigrator_test/flow2 complete	datamigrator_test	flow2_process	20
8	2015/02/19 00:37:18	(ICM18039)	DEP_0 flow2 Return Code = 0	datamigrator_test	flow2_process	20
9	2015/02/19 00:37:18	(ICM18015)	DEP_1: procedure success started.	datamigrator_test	flow2_process	20
10	2015/02/19 00:37:18		SUCCESS!	datamigrator_test	flow2_process	20
11	2015/02/19 00:37:18	(ICM18039)	DEP_1 success Return Code = 0	datamigrator_test	flow2_process	20
12	2015/02/19 00:37:18	(ICM18031)	Finished	datamigrator_test	flow2_process	20
13	2015/02/19 00:37:18	(ICM18072)	Elapsed run time 0:00:48	datamigrator_test	flow2_process	20
14	2015/02/19 00:37:19	(ICM18975)	------ End of Log Record for datamigrator_test/flow2_process ------	datamigrator_test	flow2_process	20

Figure 84: Log from process flow run

The log shows the message:

SUCCESS!

Which means it was generated by the stored procedure called **success**.

Note: If you click the highlighted line (Link to log), you will see the detail log of that step.

This was a pretty simple process flow, but it demonstrates the potential for more complex flows. For example, the process flow below shows some of range of possibilities with a process flow.

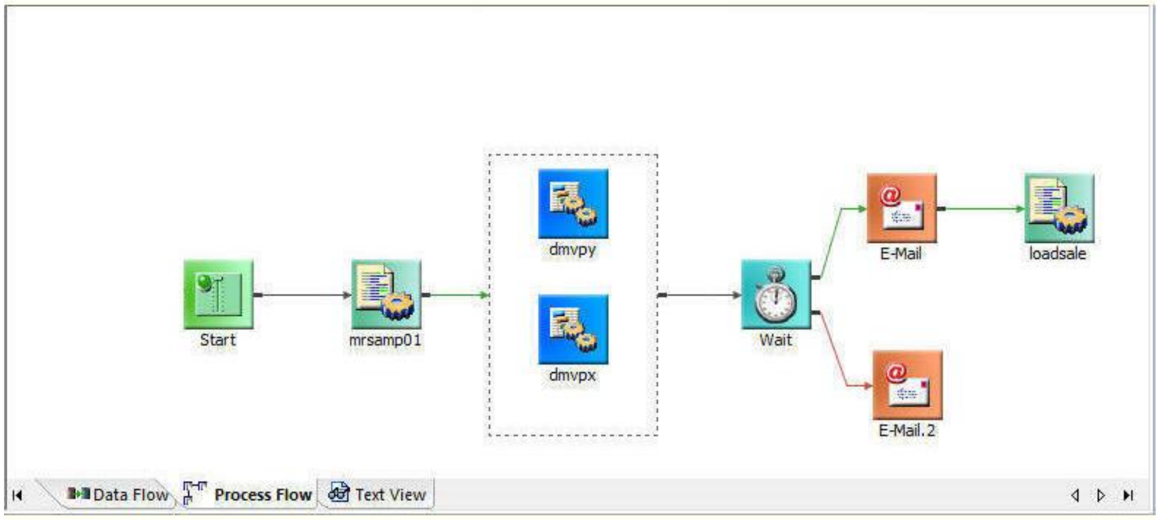


Figure 85: Example of a more complicated process flow using email notification

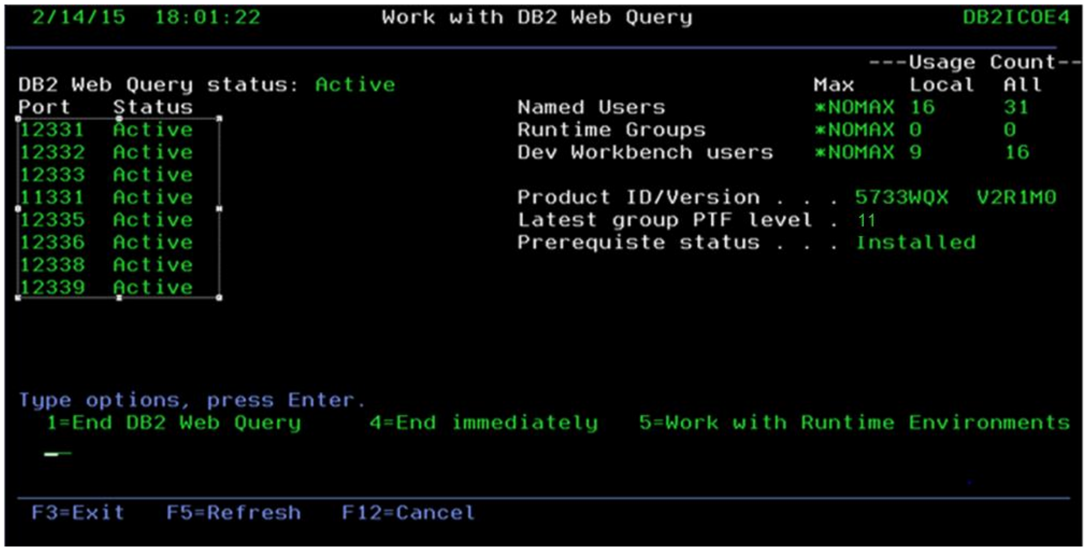
Chapter 6: Scheduling and e-mail

Overview

When performing ETL, an important requirement for the ETL process is to run in a recurring, automated fashion. One common requirement is for the ETL process to run once a day, normally at night, to capture the days changes and incorporate them into a data warehouse. Another important feature is getting some sort of confirmation, usually via email, that the ETL process was successful or at least get notified if a problem occurred.

DataMigrator for i provides a scheduling ability. This scheduling tool is separate from Web Query's report broker, allowing it to run independently of any scheduled reports. In addition, DataMigrator for i provides email notification for key steps in the ETL process, such as on the success or failure of a process flow to complete.

> **Note**: in order for scheduling to work, the DataMigrator for i server, which is actually the Web Query server, must be active. You can check if the DB2 Web Query server is active by using the CL command WRKWEBQRY to verify the status is Active and all ports are active.



```
 2/14/15  18:01:22         Work with DB2 Web Query                 DB2ICOE4

                                                            ---Usage Count--
DB2 Web Query status: Active                            Max     Local   All
Port    Status                    Named Users           *NOMAX 16       31
12331   Active                    Runtime Groups        *NOMAX 0        0
12332   Active                    Dev Workbench users   *NOMAX 9        16
12333   Active
11331   Active                    Product ID/Version . . . 5733WQX  V2R1M0
12335   Active                    Latest group PTF level . 11
12336   Active                    Prerequiste status . . . Installed
12338   Active
12339   Active



Type options, press Enter.
  1=End DB2 Web Query     4=End immediately   5=Work with Runtime Environments
  _

 F3=Exit   F5=Refresh   F12=Cancel
```

Figure 86: WRKWEBQRY screen showing all ports 12331-12339 as Active

Scheduling setup

Schedule setup is handled through the DMC. To schedule a flow, right click on the flow and select *Schedule and E-mail* → *Manage*. Do this now for flow **flow2**.

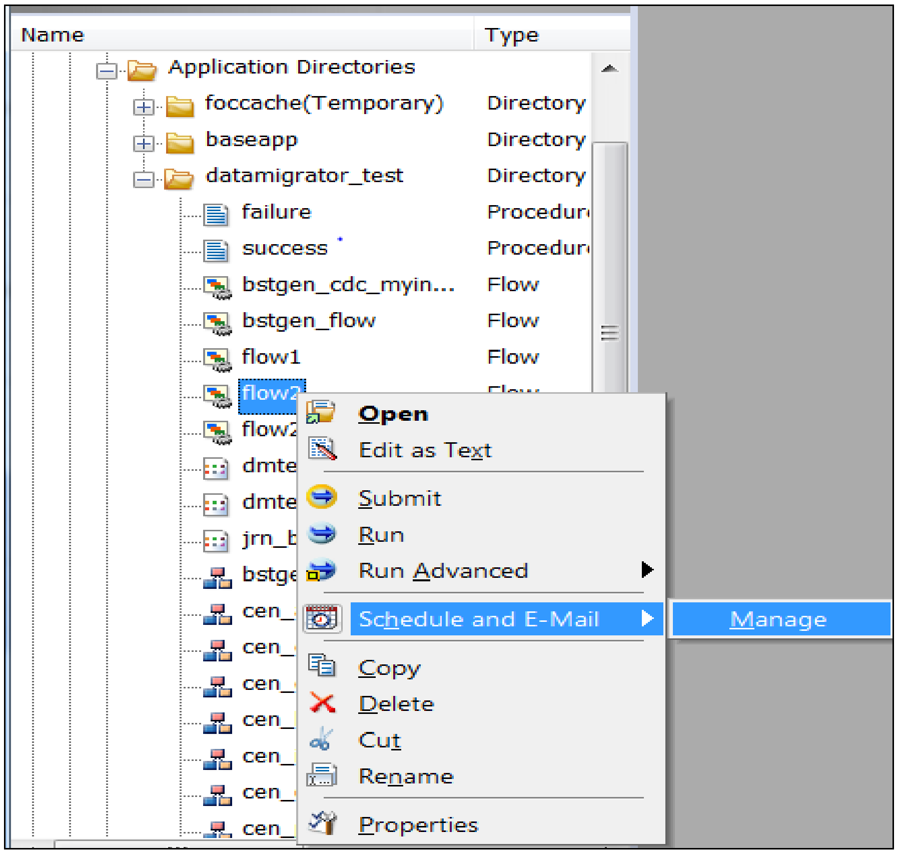
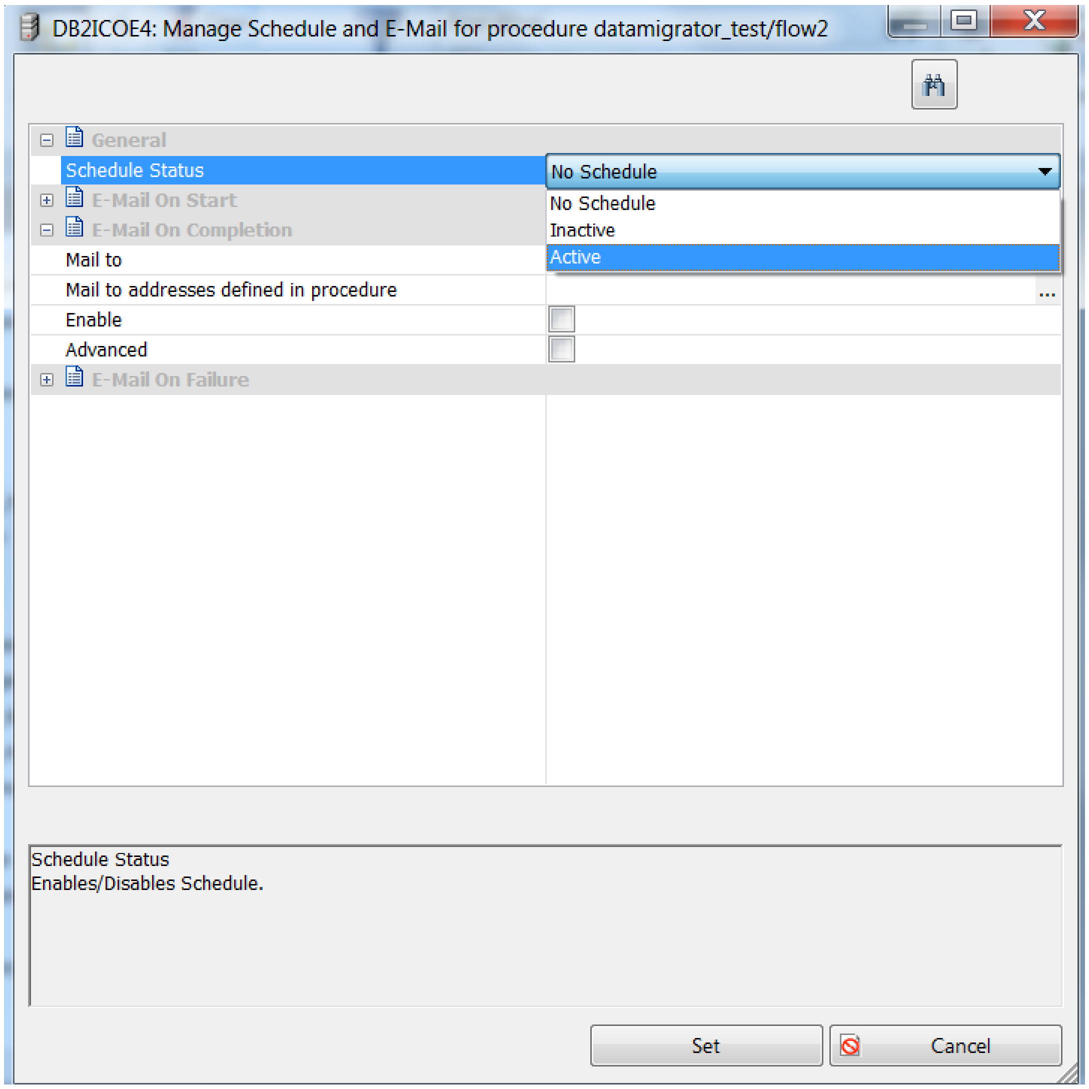


Figure 86a: Scheduling a flow

From the scheduling screen we can setup the scheduling details for the flow. Choose *Active* from the Schedule Status dropdown.


DB2ICOE4: Manage Schedule and E-Mail for procedure datamigrator_test/flow2

General	
Schedule Status	No Schedule
E-Mail On Start	
E-Mail On Completion	
Mail to	
Mail to addresses defined in procedure	...
Enable	☐
Advanced	☐
E-Mail On Failure	

Schedule Status
Enables/Disables Schedule.

Figure 87: Setting a schedule to Active

Once the *Schedule Status* is *Active, a Scheduling Type* option appears. Several options are available and are fairly self explanatory: *Run Once*, *Recurring*, *Multi-Day*, and *Run when server starts*.

Select *Multi-Day* from the dropdown. For *Start and stop range*, the *Start Date* and *Start Time* are pre-populated with the current date and time. Let's change the start time to be Midnight or 00:00. While we could specify an ending date, we will leave it open ended.

Under *Special date/time ranges* we can choose from multiple different days of the week and days of the month. We still setup to run every other day of the week, on the 15th of each month (middle of the month) and also on the last day of the month.

Select the ellipsis (…) on *Days of the Week*. This will bring up a window with days of the week. To select a day simply click on it. To deselect a selected day simply click it again. Select **Sunday, Tuesday, Thursday** and **Saturday**.

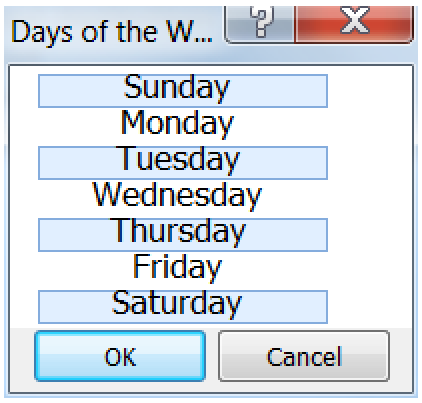


Figure 88: Days of the week selected

Select OK.

Click on the *Days of the Month* ellipsis (…). This will bring up a window with days of the month and a special value *Last Day of Month*. As with days of the week, to select a day simply click on it. To deselect a selected day simply click it again. Select *15* and *Last Day of Month*.

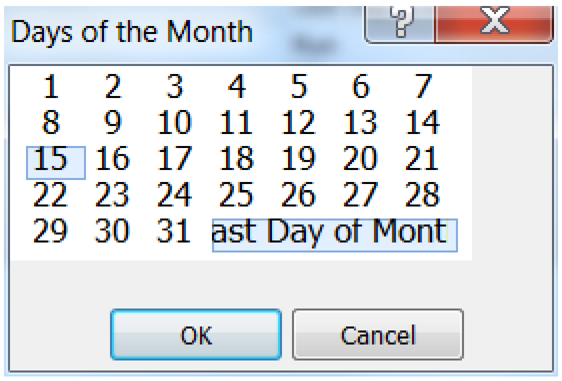
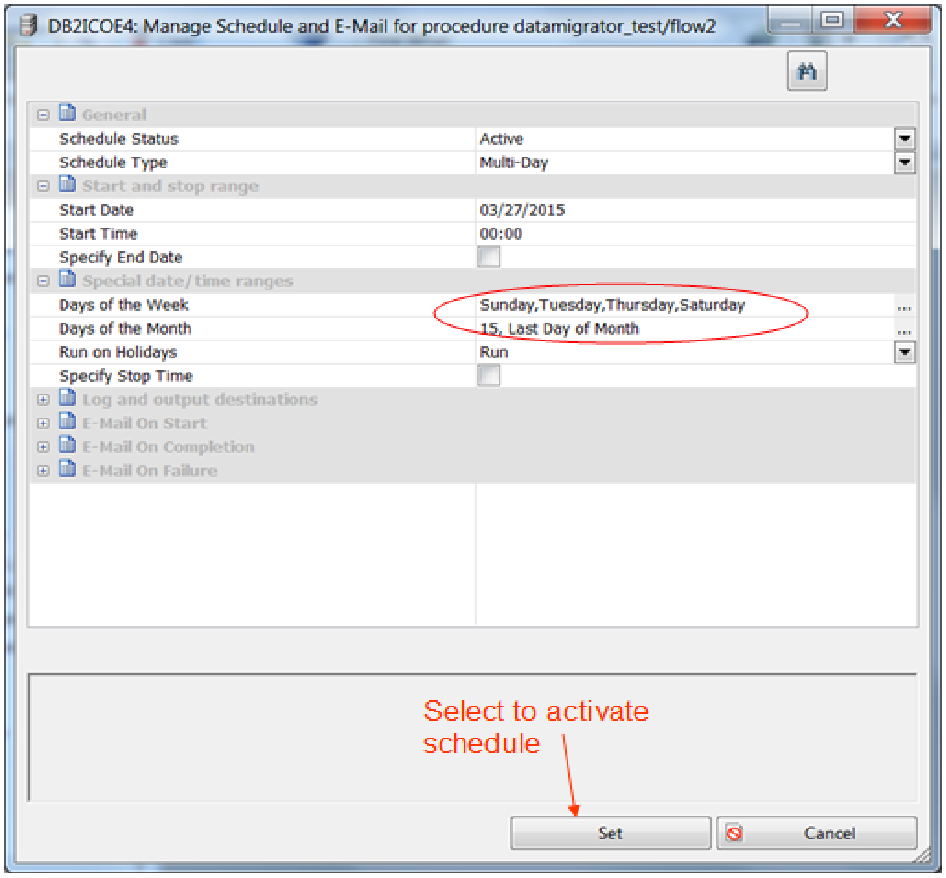


Figure 89: Days of the month selected

Select OK.

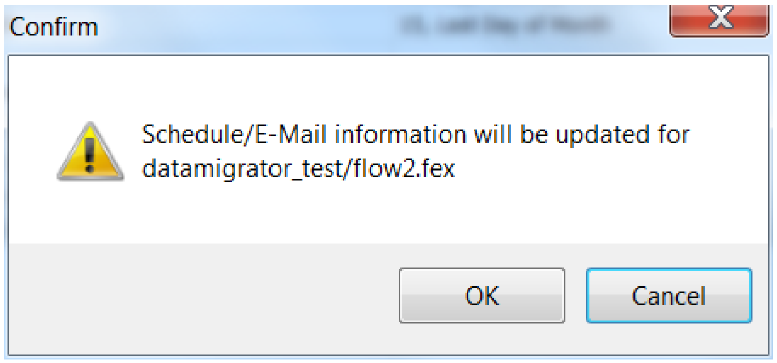
Back on the main schedule screen, verify the scheduled dates are shown and then select *Set* to activate the schedule.


DB2ICOE4: Manage Schedule and E-Mail for procedure datamigrator_test/flow2

General	
Schedule Status	Active
Schedule Type	Multi-Day
Start and stop range	
Start Date	03/27/2015
Start Time	00:00
Specify End Date	☐
Special date/time ranges	
Days of the Week	Sunday,Tuesday,Thursday,Saturday
Days of the Month	15, Last Day of Month
Run on Holidays	Run
Specify Stop Time	☐
Log and output destinations	
E-Mail On Start	
E-Mail On Completion	
E-Mail On Failure	

Figure 90: Verify scheduled dates and activate with Set button

A verification screen is displayed.


Confirm

Schedule/E-Mail information will be updated for datamigrator_test/flow2.fex

Figure 91: schedule enablement verification

Select OK.

Looking closely at the icon associated with flow2 in the left navigation tree, we see that it has changed to a clock, indicating it is a scheduled flow.

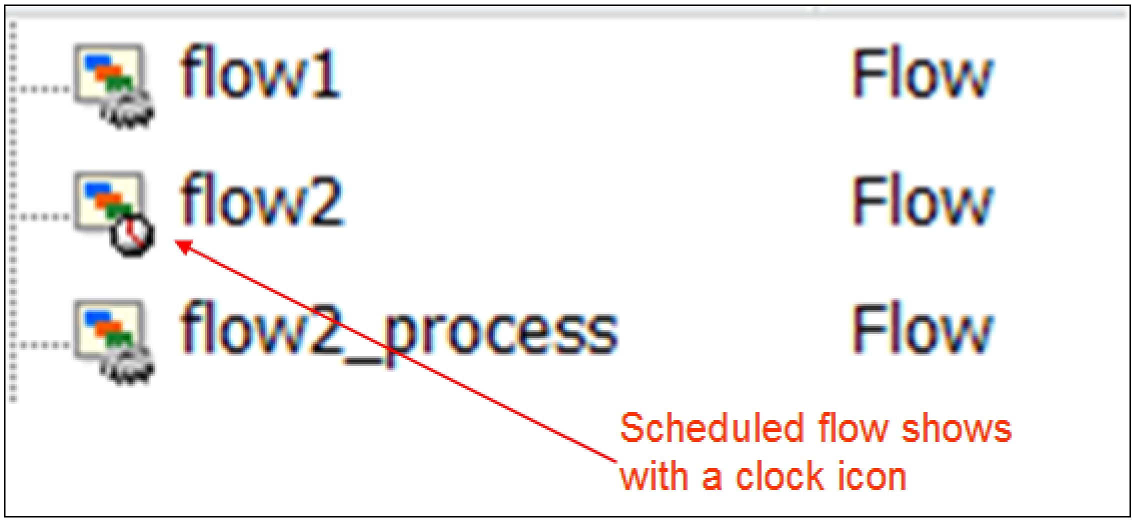


Figure 92: Scheduled flow showing a clock icon

At this point flow2 is scheduled to run.

E-mail notification setup

DataMigrator for i provides several built-in e-mail options for scheduling. An email can be sent when a scheduled flow starts, when it completes, and/or when it fails. Depending on your needs, you may want to know every time it starts and runs successfully or only when it fails. We will setup to only be informed when the scheduled flow fails.

Bring up the scheduling screen again by right clicking on flow2 and selecting *Schedule and E-mail* → *Manage*

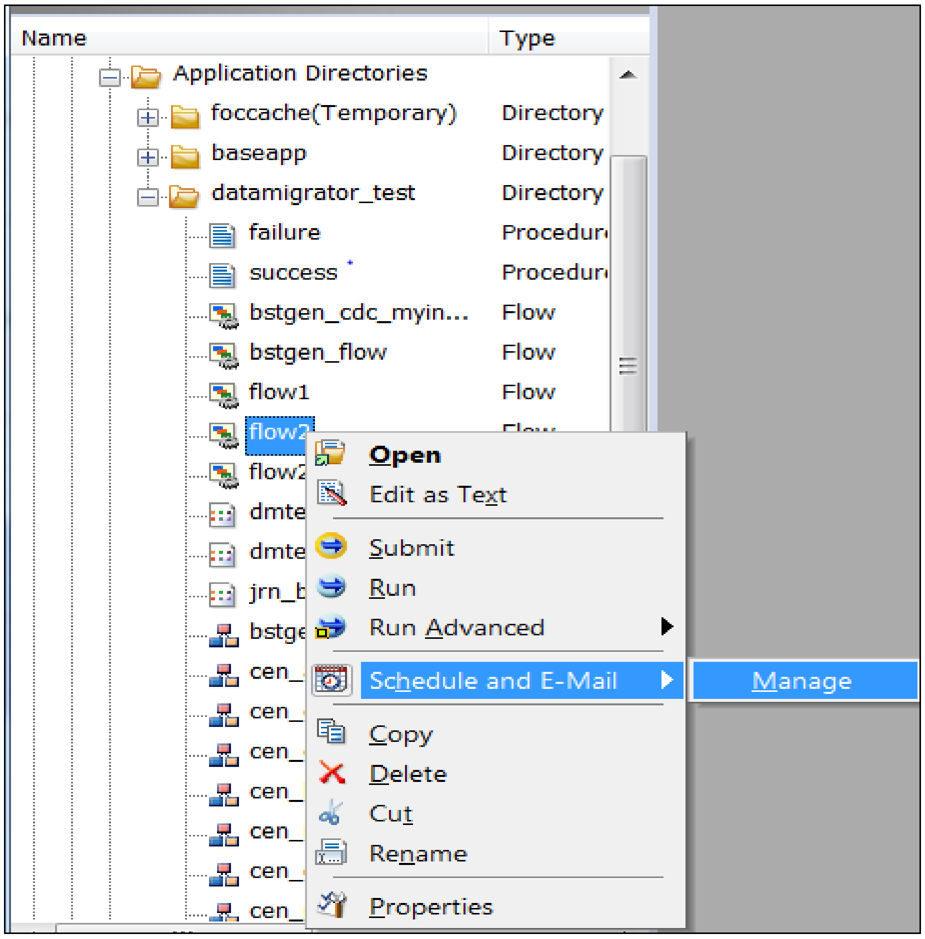
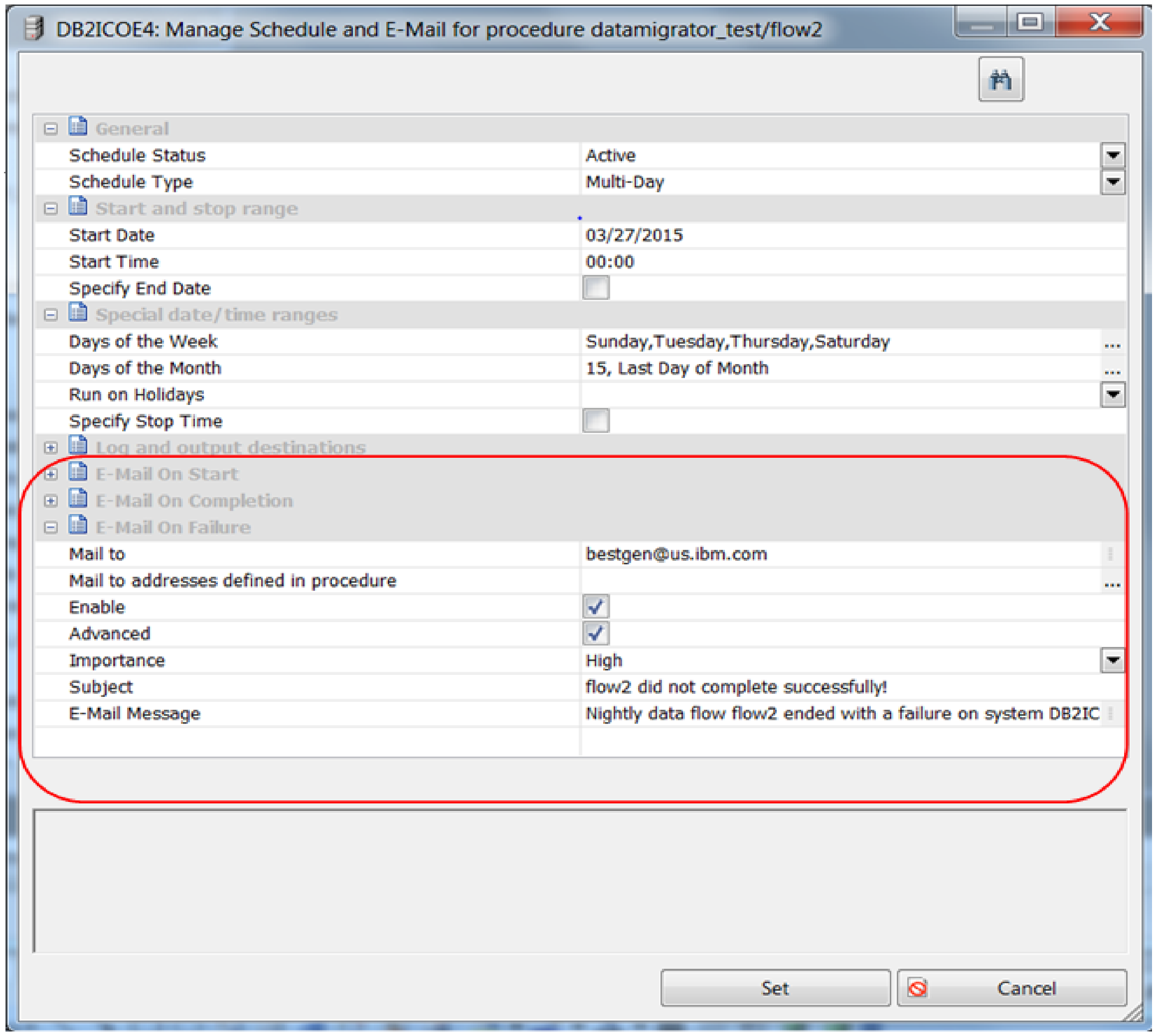


Figure 93: Manage an active schedule for a flow

E-mail control options are shown at the bottom of the screen. Expand the E-mail on failure section. For the *Mail to* option, click on the expand option on the right to show a text area. Type in your email address. Next, enable the email option by checking the box next to *Enable*. Then check the box next to the *Advanced* option. This will expand to show options for setting the email Importance (High, Normal or Low), a subject line for the email and the email body. Fill in the following:

- Set the *Importance* to *High*
- Set e-mail Subject to 'flow2 did not complete successfully!'.
- Set e-mail message to 'Nightly data flow flow2 ended with a failure on system xxxx' where xxxx is your system name.



DB2ICOE4: Manage Schedule and E-Mail for procedure datamigrator_test/flow2

General	
Schedule Status	Active
Schedule Type	Multi-Day
Start and stop range	
Start Date	03/27/2015
Start Time	00:00
Specify End Date	☐
Special date/time ranges	
Days of the Week	Sunday,Tuesday,Thursday,Saturday
Days of the Month	15, Last Day of Month
Run on Holidays	
Specify Stop Time	☐
Log and output destinations	
E-Mail On Start	
E-Mail On Completion	
E-Mail On Failure	
Mail to	bestgen@us.ibm.com
Mail to addresses defined in procedure	
Enable	✓
Advanced	✓
Importance	High
Subject	flow2 did not complete successfully!
E-Mail Message	Nightly data flow flow2 ended with a failure on system DB2IC

Figure 94: E-mail on failure setup

Verify everything looks ok and select *Set* to activate. Select OK on the confirmation screen.

We now have flow2 set to run four days a week, plus on the 15th and last day of each month, and to send us an email if something fails.

Warning: Since this was a test flow, you likely don’t really want it to run on your system. Therefore, before leaving this chapter, bring up the schedule again and change the *Schedule Status* to *Inactive*. This will preserve all the schedule settings but will avoid actually running the flow.

Chapter 7: Using Change Data Capture (CDC)

Overview

Change Data Capture (CDC) enables reading from **journals** to determine what rows in an associated table (file) have changed. This approach is very useful for capturing maintenance changes or 'trickle feed' ETL from large source tables where it would take too long to process the entire tables.

To use Change Data Capture, the file must be configured to use journaling. A *Table Log Records* synonym is created to represent a log 'table', allowing reading from the journal entries and having them represented as records from the selected table. In other words, the Table Log Records synonym is really a synonym for a journal but has columns that match the associated table being journaled. This table log synonym can be used as a source in a data flow or in a direct load flow.

An important aspect to understand about DataMigrator for i CDC is that it works on a transaction basis. When processing entries from a journal, CDC only processes transactions that have been committed. For example, if a database job is writing records to a journaled table and CDC is setup against the table's journal, none of the records written into the table (and therefore the journal) will be processed by CDC until the database job issues a COMMIT. Once the COMMIT entry makes it to the journal CDC will then consider reading the record images from the journal and processing them.

Definitions

The first four columns in a log synonym are added automatically by DataMigrator for i to every Table Log Record synonym. They are used for flow control during DataMigrator for i IUD processing. The columns have the following functions:

- **CDC_OPER -** the operation type (Insert, Update, or Delete)
- **CDC_TID -** the Transaction ID
- **CDC_TIMES -** the time stamp
- **CDC_RRN -** RRN of the record

The Configuration section of the CDC synonym also consists of special parameters with properties that can be customized by a user. Each CDC parameter in the configuration section of the CDC synonym has a default value of a global variable. Each global variable name begins with **&&CDC_** followed by the parameter name.

The following options are available:

DB Log Parameters

- *DATA_ORIGIN*

  This is set to the value **DBMSLOG** to indicate the origin of the data is a DBMS table log.

- *START*

  Indicates the starting point for reading journal entries, known as ‘log records’. Possible values:

  - **CHKPT -** After the last transaction retained in the checkpoint file. This is the default.
  - **CUR_TRAN -** The first transaction in the journal after the job started.
  - **CUR_LOG -** The first available transaction in the current active journal receiver.

- *CHKPT_SAVE*

  Indicates if the last processed transaction should be saved in a checkpoint file. Possible values:

  - **YES -** Retain the last processed transaction in the checkpoint file.
  - **NO -** Do not retain the last processed transaction. This option facilitates testing, because using it during Sample Data or Test Transforms will not reset the checkpoint.

- *CHKPT_FILE*

  Indicates the location and name of the file to use to store checkpoint information.

  - **Physical location.** The full directory and file name in the format for the operating system used. This option should be used when there are multiple flows that will access the table log synonym for the same table. Each flow should have a unique synonym with a unique name for the checkpoint file.
  - **Application directory.** The file is created in the same application directory as the synonym.
  - **Blank.** This is the default. If no file name is specified, a file is created with the same name as the synonym and the extension *chp*.

- *COMMIT_MODE*

  Supports the transactional commitment control.

- **ON.** Normal transaction mode with a commit used. This is the default.
- **OFF.** No commit issued or DBMS uses auto-commit mode.
- **DTC.** Distributed transactions mode is used.

➢ *LOG_NAME*

Indicates the name of the journal. If this is blank DataMigrator for i defaults to QSQJRN, which is the name of the journal associated with SQL tables.

➢ *LOG_LOCATION*

Indicates the journal library. If this is blank DataMigrator for i defaults to the same library as the associated database table.

Listening Parameters

➢ *POLLING*

Indicates the polling interval in seconds, or how often the database log is scanned. The default is 1 second.

➢ *TIMEOUT*

Indicates the timeout interval in seconds. If there is no activity in this time interval, the processing will stop. A value of zero means there is no limit. The default value is 1 second.

Note that the two options above work together. For example, if POLLING is 2 and TIMEOUT is 10, the server polls the log every 2 seconds for new transactions. If there are no new transactions for a 10 consecutive second timespan, then polling stops.

Read Limits

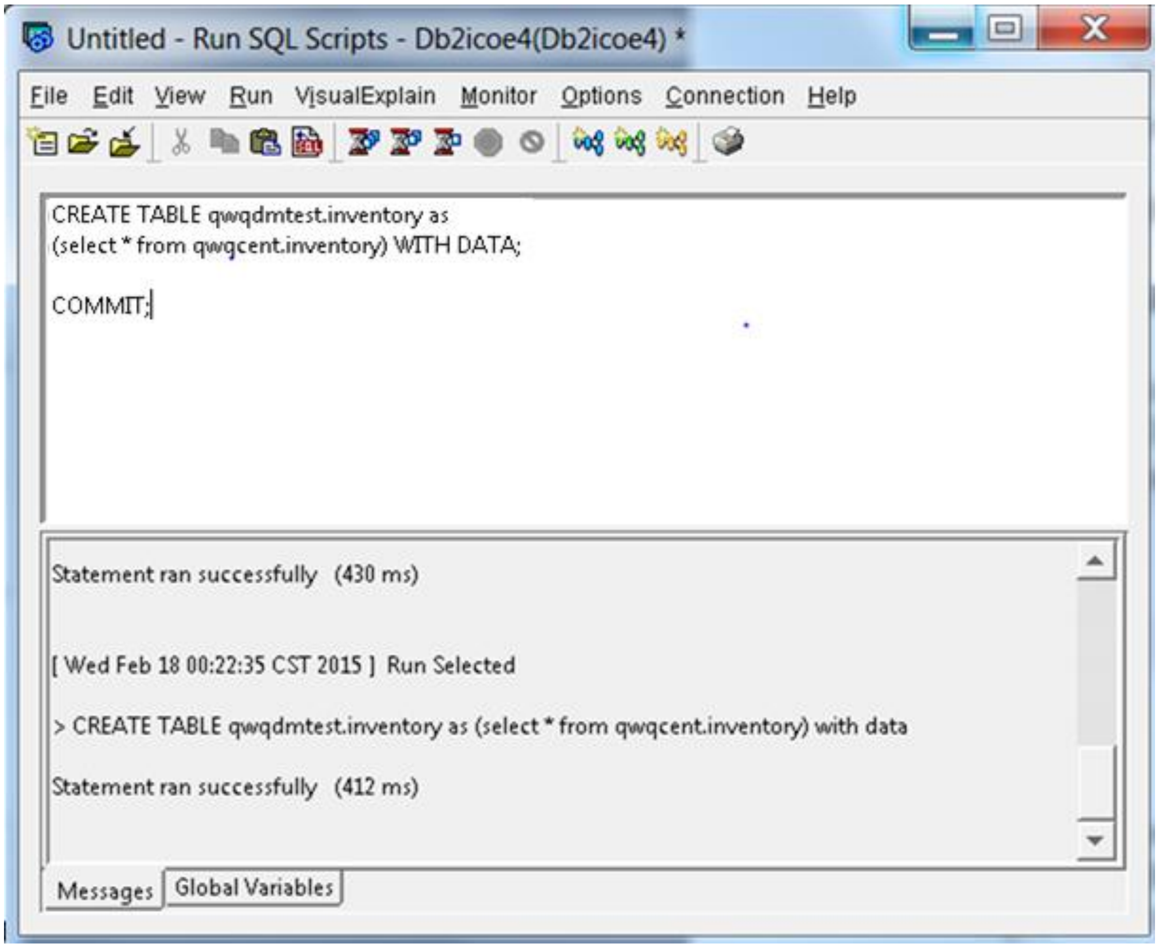
➢ *MAXLUWS*

Indicates the maximum number of database transactions (Logical Units of Work or LUWS) to process before stopping the job. A value of zero (0) means all transactions. The default value is 1 transaction.

Setup

We will need a table to use as a data source that can also be changed by adding new records. Let's copy the inventory table from QWQCENT into the QWQDMTEST library. Using SQL, issue the following statements. Make sure to be running under some level of commitment control, for example *CHG.

```
CREATE TABLE qwqdmtest.inventory as
(select * from qwqcent.inventory) WITH DATA;

COMMIT;
```


Untitled - Run SQL Scripts - Db2icoe4(Db2icoe4) *
File Edit View Run VisualExplain Monitor Options Connection Help
Messages Global Variables

```
CREATE TABLE qwqdmtest.inventory as
(select * from qwqcent.inventory) WITH DATA;

COMMIT;
```

```
Statement ran successfully   (430 ms)

[ Wed Feb 18 00:22:35 CST 2015 ]  Run Selected

> CREATE TABLE qwqdmtest.inventory as (select * from qwqcent.inventory) with data

Statement ran successfully   (412 ms)
```

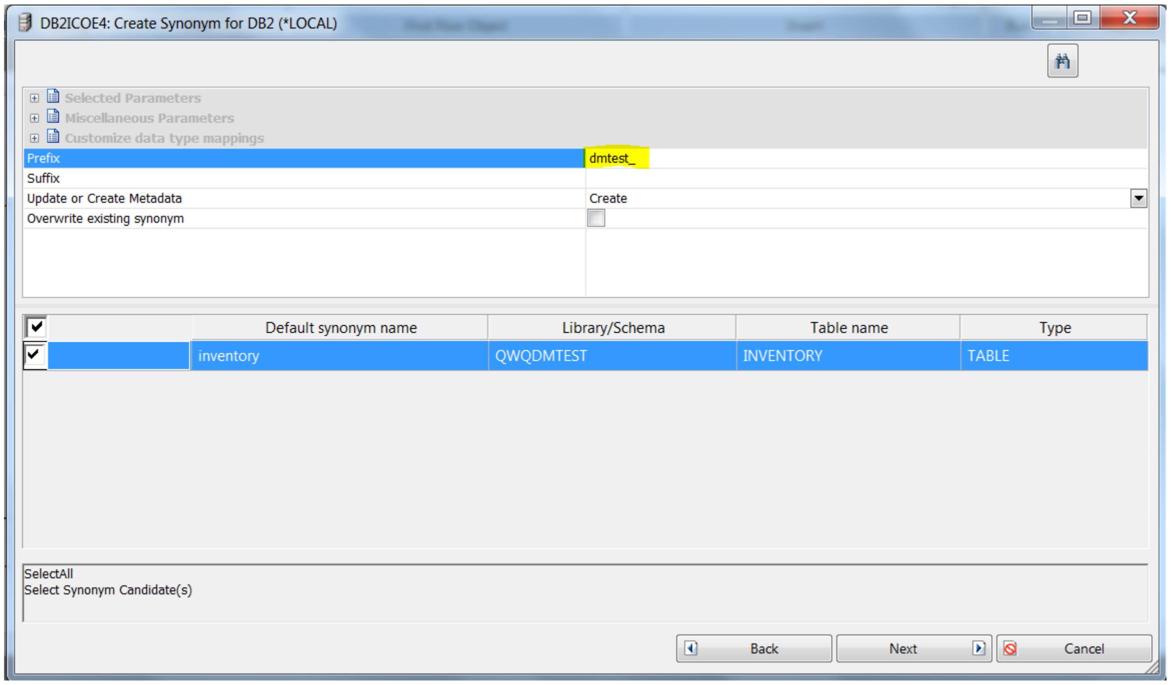
Figure 95: Create table for inventory

Note: To run under commitment control through System i Navigator's Run SQL Scripts, select *Connection* → *JDBC Settings* and change *Isolation level*.

A synonym for a journal allows reading from the journal associated with a selected table. The synonym contains the same columns as the table itself plus four additional columns that identify changes to the table.

However, a synonym for a journal is **not** a substitute for a synonym for the table itself. A synonym is still needed for the table to be able to read or write from the table directly. So we need two synonyms.

First, create a normal synonym for our new **qwqdmtest.inventory** table. Give it a prefix of **dmtest_**. Remember to specify adapter of *LOCAL.


DB2ICOE4: Create Synonym for DB2 (*LOCAL)
Selected Parameters
Miscellaneous Parameters
Customize data type mappings
SelectAll
Back
Next
Cancel

Prefix	dmtest_
Suffix	
Update or Create Metadata	Create
Overwrite existing synonym	☐

✓	Default synonym name	Library/Schema	Table name	Type
✓	inventory	QWQDMTEST	INVENTORY	TABLE

Select Synonym Candidate(s)

Figure 96: Synonym for qwqdmtest.inventory with prefix of dmtest_

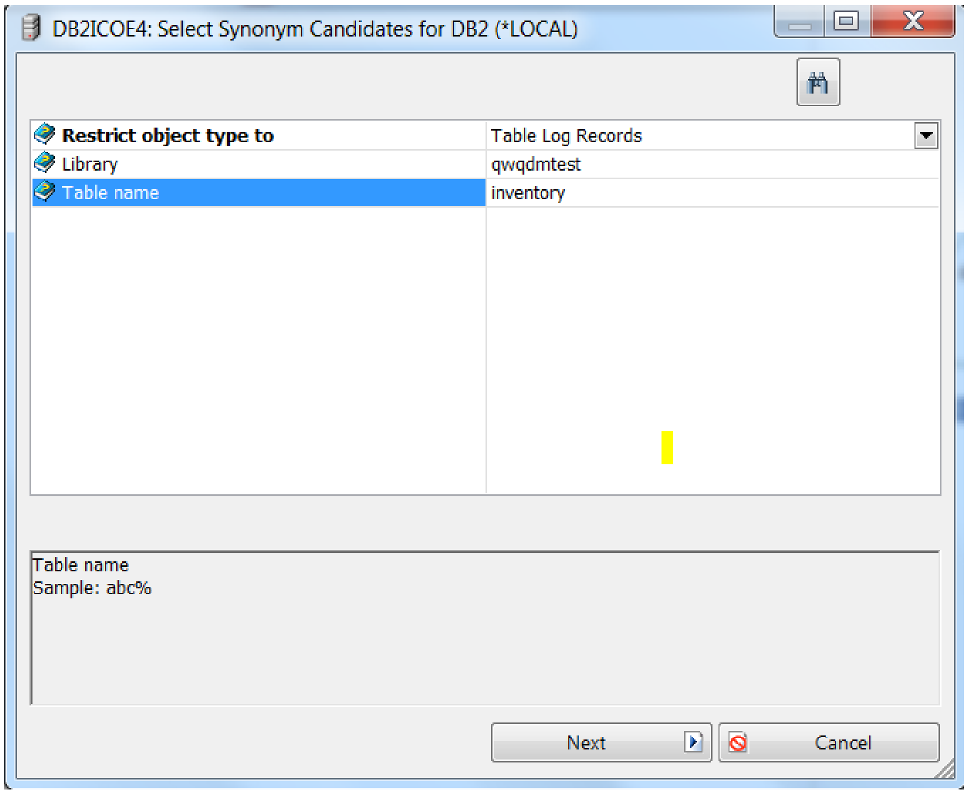
Next we need to create the *Table Log Records* synonym for the journal.

1. Right-click an application directory in the navigation pane and choose *New* and then *Synonym (Create or Update).*

   The Select Adapter window opens. Choose *LOCAL.

2. From the *Restrict object type to* drop-down menu, select *Table Log Records*.

   Specify Library **qwqdmtest** and Table name **inventory**.



DB2ICOE4: Select Synonym Candidates for DB2 (*LOCAL)

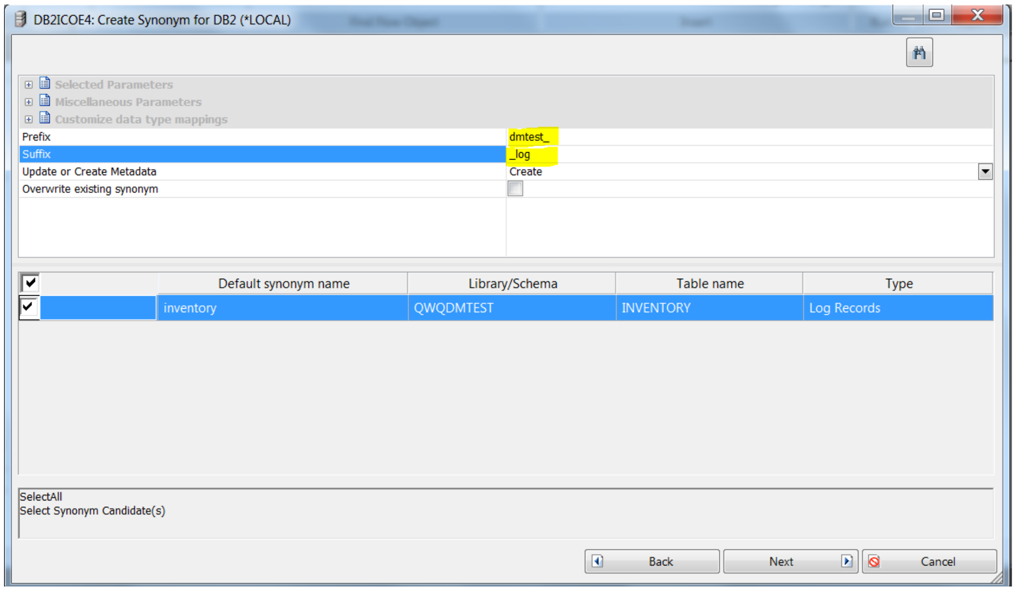
Restrict object type to	Table Log Records
Library	qwqdmtest
Table name	inventory

Table name
Sample: abc%

Figure 97: Table log synonym for qwqdmtest.inventory

3. Click *Next*.
4. Select the check box in front of **inventory**.

   For the synonym prefix, specify **dmtest_** like we did for the table. In addition, for the suffix specify **_log**. This suffix will be a reminder that this synonym reads from a log (journal).


DB2ICOE4: Create Synonym for DB2 (*LOCAL)

Selected Parameters
Miscellaneous Parameters
Customize data type mappings

Prefix	dmtest_
Suffix	_log
Update or Create Metadata	Create
Overwrite existing synonym	☐

✓	Default synonym name	Library/Schema	Table name	Type
✓	inventory	QWQDMTEST	INVENTORY	Log Records

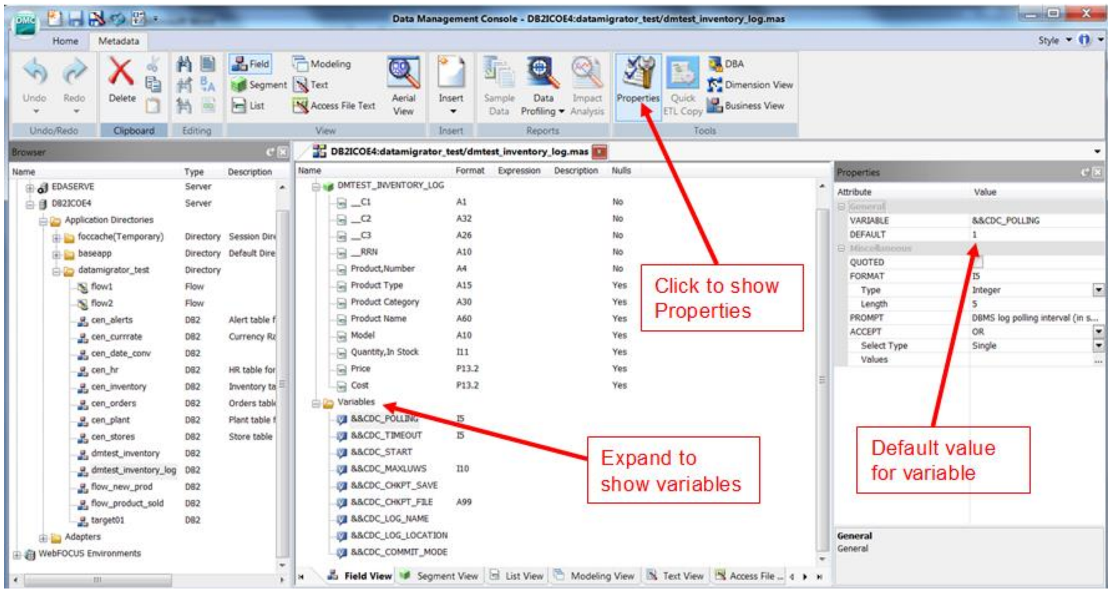
SelectAll
Select Synonym Candidate(s)

Figure 98: Table log prefix dmtest_ and suffix _log

5. Click *Next* to create the Table Log Record synonym.
6. Click *Close* on the resulting status screen.

Let’s take a look at the synonym we just created. Open the newly created **dmtest_inventory_log** synonym. Then open the *Properties* for the synonym. Finally, expand the *Variables* folder to show the variables. This view shows all the columns, including the automatically added ones at the beginning, plus the global variables used to set the options described earlier.

Notice that the automatically added columns have a short name _Cx, for example _C1. However, selecting any column shows the long name in the properties. Examine the contents of the synonym to become familiar with it.



DB2ICOE4:datamigrator_test/dmtest_inventory_log.mas

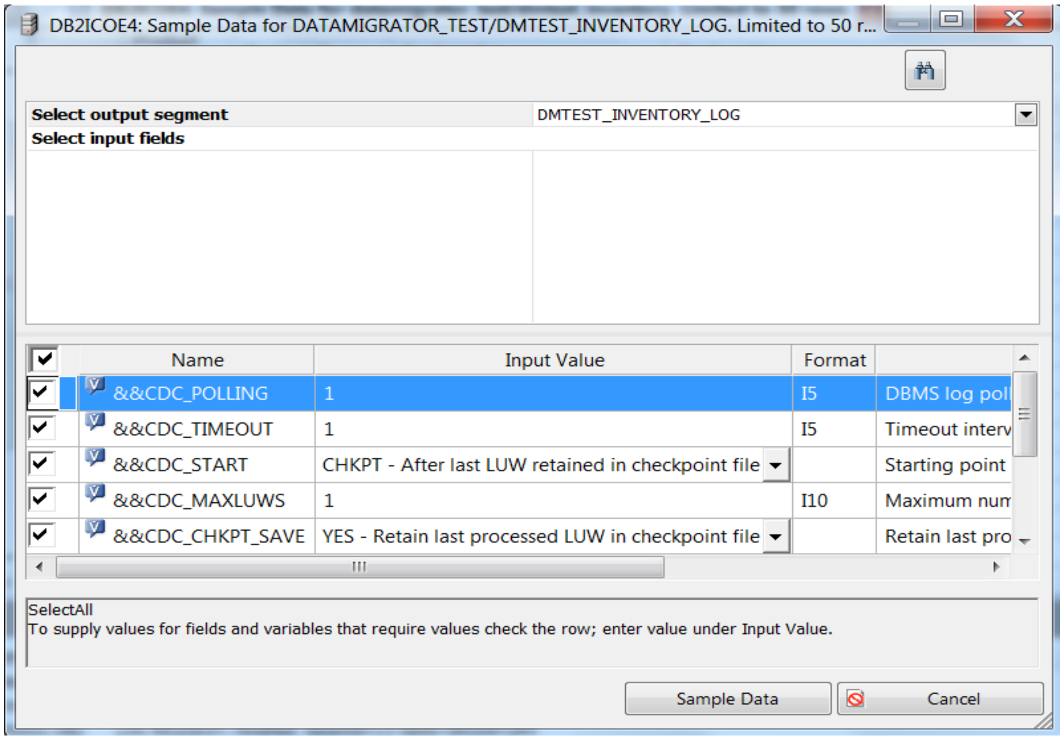
Name	Format	Expression	Description	Nulls
DMTEST_INVENTORY_LOG				
__C1	A1			No
__C2	A32			No
__C3	A26			No
__RRN	A10			No
Product,Number	A4			No
Product Type	A15			Yes
Product Category	A30			Yes
Product Name	A60			Yes
Model	A10			Yes
Quantity,In Stock	I11			Yes
Price	P13.2			Yes
Cost	P13.2			Yes
Variables				
&&CDC_POLLING	I5			
&&CDC_TIMEOUT	I5			
&&CDC_START				
&&CDC_MAXLUWS	I10			
&&CDC_CHKPT_SAVE				
&&CDC_CHKPT_FILE	A99			
&&CDC_LOG_NAME				
&&CDC_LOG_LOCATION				
&&CDC_COMMIT_MODE				

Properties

Attribute	Value
VARIABLE	&&CDC_POLLING
DEFAULT	1
QUOTED	
FORMAT	I5
Type	Integer
Length	5
PROMPT	DBMS log polling interval (in s...
ACCEPT	OR
Select Type	Single
Values	

Figure 99: View a Table Log Record synonym

Now sample the data to verify the journal is being read. Right-click on the synonym and specify *Sample Data*. On the sample data window select (checkmark) all variables.


DB2ICOE4: Sample Data for DATAMIGRATOR_TEST/DMTEST_INVENTORY_LOG. Limited to 50 r...

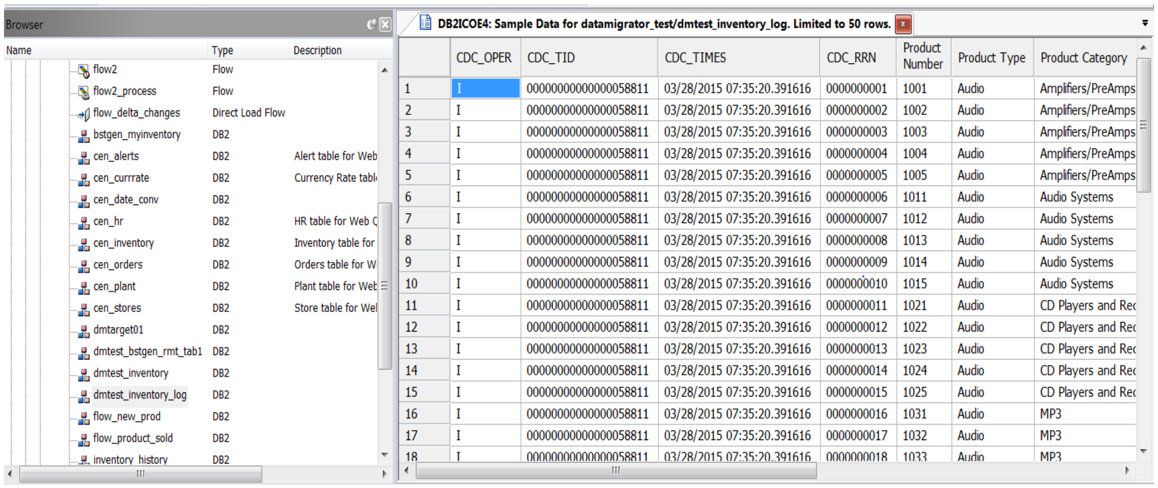
Select output segment	DMTEST_INVENTORY_LOG
Select input fields	

✓	Name	Input Value	Format	
✓	&&CDC_POLLING	1	I5	DBMS log pol
✓	&&CDC_TIMEOUT	1	I5	Timeout interv
✓	&&CDC_START	CHKPT - After last LUW retained in checkpoint file		Starting point
✓	&&CDC_MAXLUWS	1	I10	Maximum nur
✓	&&CDC_CHKPT_SAVE	YES - Retain last processed LUW in checkpoint file		Retain last pro

SelectAll
To supply values for fields and variables that require values check the row; enter value under Input Value.

Figure 100: Sample Data dialog

Click *Sample Data*



Browser

Name	Type	Description
flow2	Flow	
flow2_process	Flow	
flow_delta_changes	Direct Load Flow	
bstgen_myinventory	DB2	
cen_alerts	DB2	Alert table for Web
cen_currrate	DB2	Currency Rate tabl
cen_date_conv	DB2	
cen_hr	DB2	HR table for Web C
cen_inventory	DB2	Inventory table for
cen_orders	DB2	Orders table for W
cen_plant	DB2	Plant table for Web
cen_stores	DB2	Store table for Wel
dmtarget01	DB2	
dmtest_bstgen_rmt_tab1	DB2	
dmtest_inventory	DB2	
dmtest_inventory_log	DB2	
flow_new_prod	DB2	
flow_product_sold	DB2	
inventory_history	DB2	

DB2ICOE4: Sample Data for datamigrator_test/dmtest_inventory_log. Limited to 50 rows.

	CDC_OPER	CDC_TID	CDC_TIMES	CDC_RRN	Product Number	Product Type	Product Category
1	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000001	1001	Audio	Amplifiers/PreAmps
2	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000002	1002	Audio	Amplifiers/PreAmps
3	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000003	1003	Audio	Amplifiers/PreAmps
4	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000004	1004	Audio	Amplifiers/PreAmps
5	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000005	1005	Audio	Amplifiers/PreAmps
6	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000006	1011	Audio	Audio Systems
7	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000007	1012	Audio	Audio Systems
8	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000008	1013	Audio	Audio Systems
9	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000009	1014	Audio	Audio Systems
10	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000010	1015	Audio	Audio Systems
11	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000011	1021	Audio	CD Players and Re
12	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000012	1022	Audio	CD Players and Re
13	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000013	1023	Audio	CD Players and Re
14	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000014	1024	Audio	CD Players and Re
15	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000015	1025	Audio	CD Players and Re
16	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000016	1031	Audio	MP3
17	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000017	1032	Audio	MP3
18	I	0000000000000058811	03/28/2015 07:35:20.391616	0000000018	1033	Audio	MP3

Figure 101: Sample data from journal

You should see a sampling of the rows that were put into the journal.

> **Note**: if you don't see any sample data, it is likely that you either forgot to run the CREATE TABLE under commit or forgot to issue the subsequent COMMIT.

Close the *Sample Data tab.*

Creating Flows

With our synonyms created, it is time to create our flows. For simplicity we will create the ‘target’ tables into the same **qwqdmtest** library.

> **Note**: while putting the target tables into the same library as the source tables is possible to do, and we actually do it here for this example, in reality your target tables should be on another system from the source tables, or at least in another library on the same system.

In order to use a change data capture flow, there needs to be a base set of data in the target table. This reflects a common case for change data capture scenarios. At a minimum, two flows are used. The first flow is the ‘bulk load’ flow that is run once to populate the target tables with data to a certain point in time (usually current to the point when the flow runs). The second flow is used to capture and apply any changes that occur in the source tables after the first flow has been run.

For our scenario we will be creating and maintaining an inventory history table. This table keeps track of changing inventory on a daily basis and will allow us to track how inventory fluctuates over time.

Creating Bulk Load Flow

To build the first 'bulk load' flow, right click on the datamigrator_test folder and specify *New* → *Flow*. Drag in **dmtest_inventory** from the left navigation tree onto the left side of the flow palette. Next, click and drag *New Target* from the ribbon onto the right side of the palette.

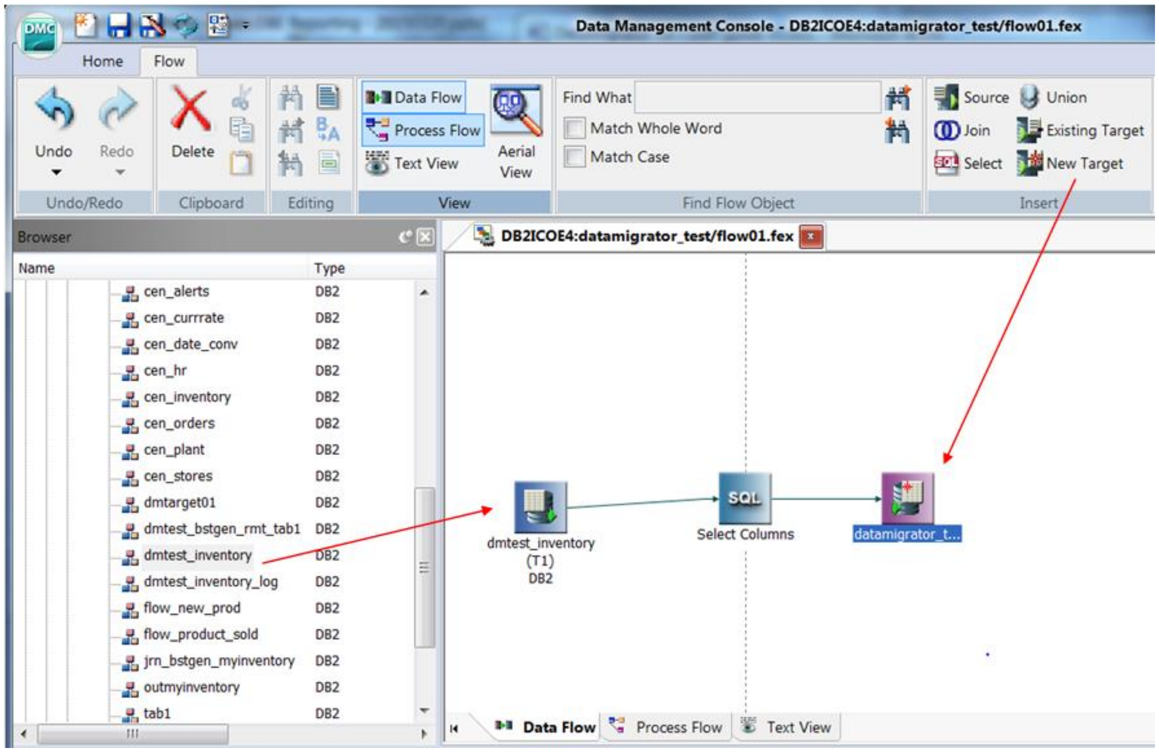
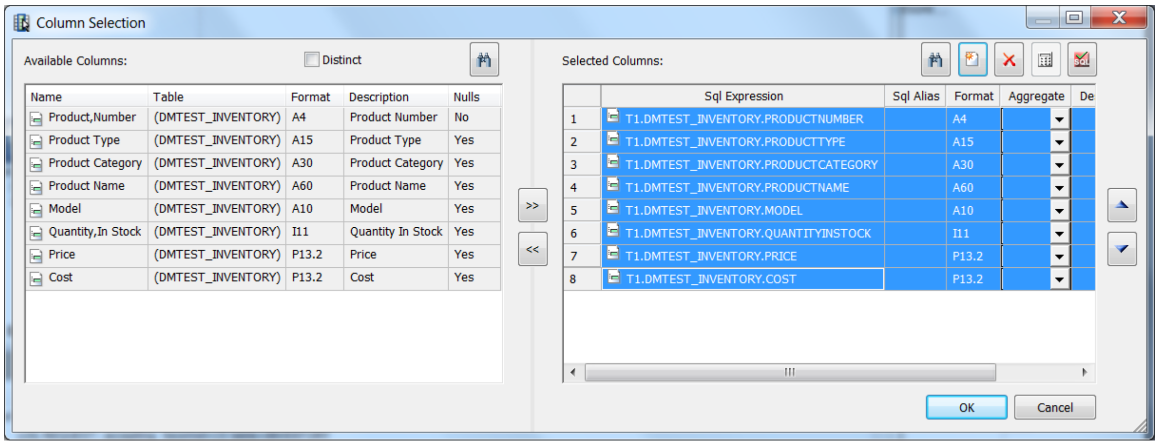


Figure 102: Create flow dragging in source and New Target

> **Note**: If you do not see *New Target* in the ribbon, widen your DMC window. The ribbon changes based on the window width and New Target could be 'hidden' because your DMC window is not wide enough to show it.

Right click on the *SQL* icon in the middle of the palette, select *Column Selection…*

Highlight all columns and click the >> arrow to select all columns.



Column Selection

Available Columns: Distinct

Name	Table	Format	Description	Nulls
Product,Number	(DMTEST_INVENTORY)	A4	Product Number	No
Product Type	(DMTEST_INVENTORY)	A15	Product Type	Yes
Product Category	(DMTEST_INVENTORY)	A30	Product Category	Yes
Product Name	(DMTEST_INVENTORY)	A60	Product Name	Yes
Model	(DMTEST_INVENTORY)	A10	Model	Yes
Quantity,In Stock	(DMTEST_INVENTORY)	I11	Quantity In Stock	Yes
Price	(DMTEST_INVENTORY)	P13.2	Price	Yes
Cost	(DMTEST_INVENTORY)	P13.2	Cost	Yes

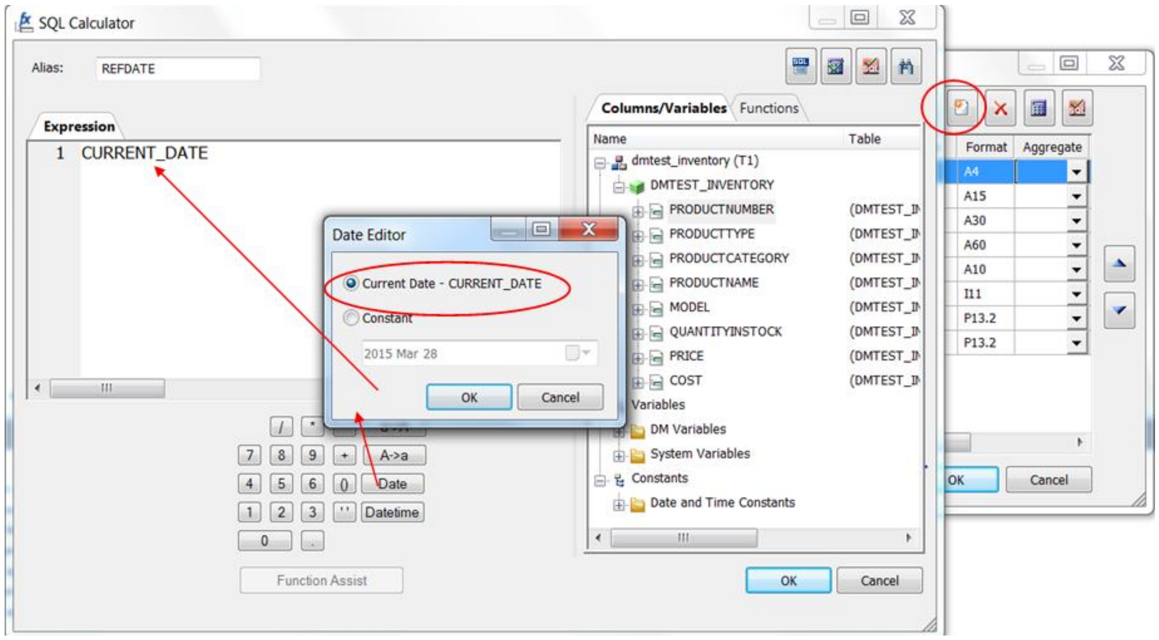
Selected Columns:

	Sql Expression	Sql Alias	Format	Aggregate	De
1	T1.DMTEST_INVENTORY.PRODUCTNUMBER		A4		
2	T1.DMTEST_INVENTORY.PRODUCTTYPE		A15		
3	T1.DMTEST_INVENTORY.PRODUCTCATEGORY		A30		
4	T1.DMTEST_INVENTORY.PRODUCTNAME		A60		
5	T1.DMTEST_INVENTORY.MODEL		A10		
6	T1.DMTEST_INVENTORY.QUANTITYINSTOCK		I11		
7	T1.DMTEST_INVENTORY.PRICE		P13.2		
8	T1.DMTEST_INVENTORY.COST		P13.2		

Figure 103: Highlight all columns and select using >> button

Since we are creating an inventory history, we need to add a date column. Click the *Insert Columns* button in the upper right to add the new column.

- Call the new column **REFDATE**.
- For the Expression click the *Date* button and select *Current Date – CURRENT DATE*.



SQL Calculator

Alias: REFDATE

Expression

```
1  CURRENT_DATE
```

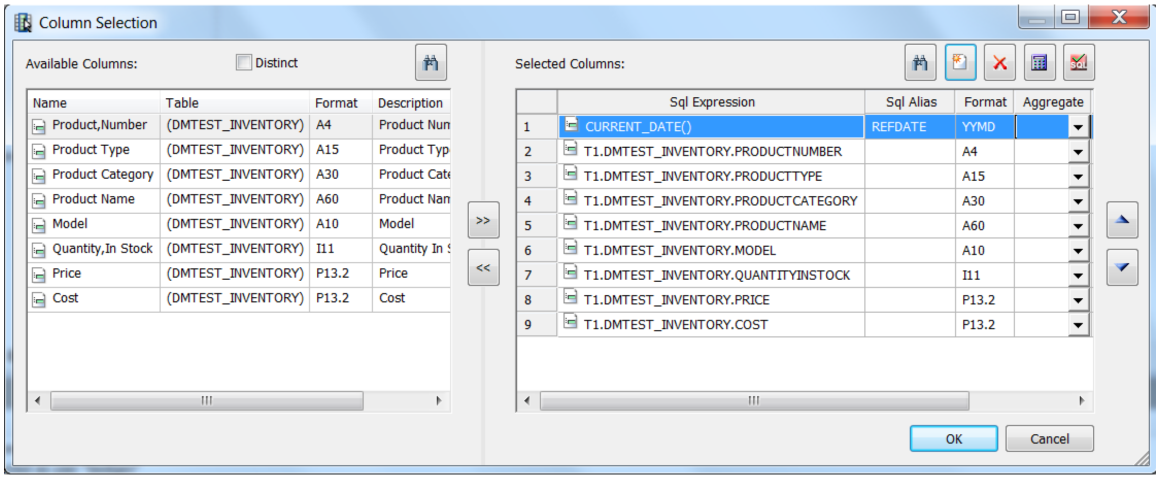
Date Editor

(● Current Date - CURRENT_DATE) ○ Constant 2015 Mar 28

Figure 104: Insert Column, new name REFDATE, Date Current Date

Select OK to create the new column.

Move the new **REFDATE** column to the top of the Select Columns list by highlighting it and clicking the up arrow key until it is at the top.


Column Selection

Available Columns: Distinct

Name	Table	Format	Description
Product,Number	(DMTEST_INVENTORY)	A4	Product Nun
Product Type	(DMTEST_INVENTORY)	A15	Product Typ
Product Category	(DMTEST_INVENTORY)	A30	Product Cate
Product Name	(DMTEST_INVENTORY)	A60	Product Nan
Model	(DMTEST_INVENTORY)	A10	Model
Quantity,In Stock	(DMTEST_INVENTORY)	I11	Quantity In S
Price	(DMTEST_INVENTORY)	P13.2	Price
Cost	(DMTEST_INVENTORY)	P13.2	Cost

Selected Columns:

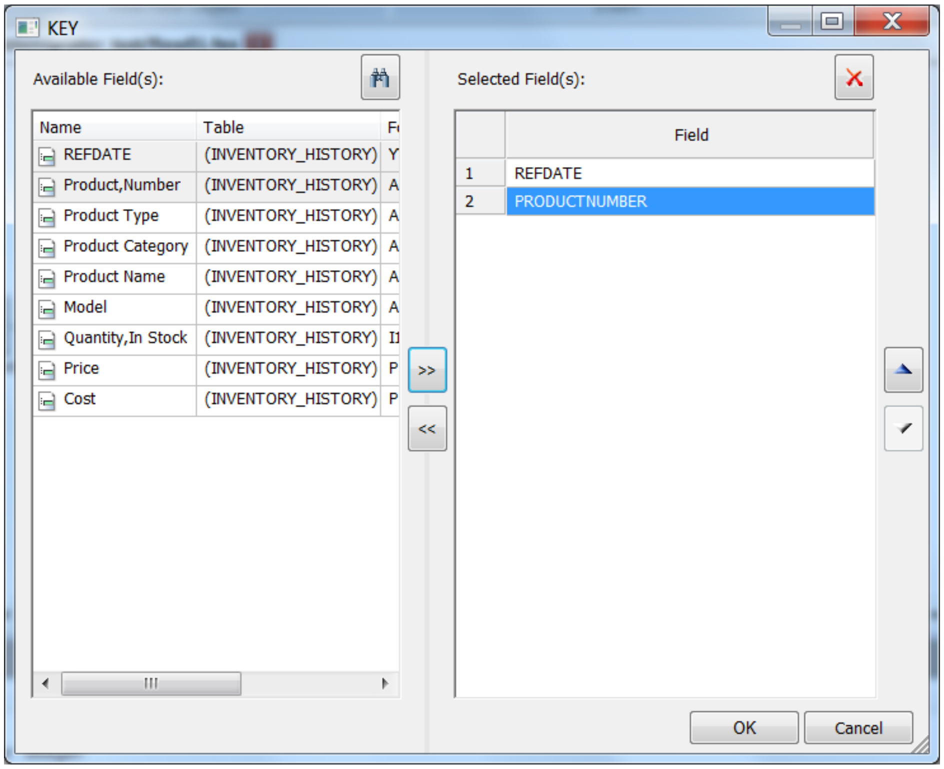
	Sql Expression	Sql Alias	Format	Aggregate
1	CURRENT_DATE()	REFDATE	YYMD	
2	T1.DMTEST_INVENTORY.PRODUCTNUMBER		A4	
3	T1.DMTEST_INVENTORY.PRODUCTTYPE		A15	
4	T1.DMTEST_INVENTORY.PRODUCTCATEGORY		A30	
5	T1.DMTEST_INVENTORY.PRODUCTNAME		A60	
6	T1.DMTEST_INVENTORY.MODEL		A10	
7	T1.DMTEST_INVENTORY.QUANTITYINSTOCK		I11	
8	T1.DMTEST_INVENTORY.PRICE		P13.2	
9	T1.DMTEST_INVENTORY.COST		P13.2	

Figure 105: Move REFDATE to the top by highlighting it and using the up arrow

Select OK.

Now we need to identify the target table. Right click on the target table icon and select *Properties*. In properties make the following changes.

- Click on the ellipsis (…) to the right of the *Keys* entry. Select **REFDATE** and **PRODUCTNUMBER** and select OK.
- For the *Table*, specify **QWQDMTEST/inventory_history**.
- For *Load Type*, choose *Insert Records from Memory*.


KEY

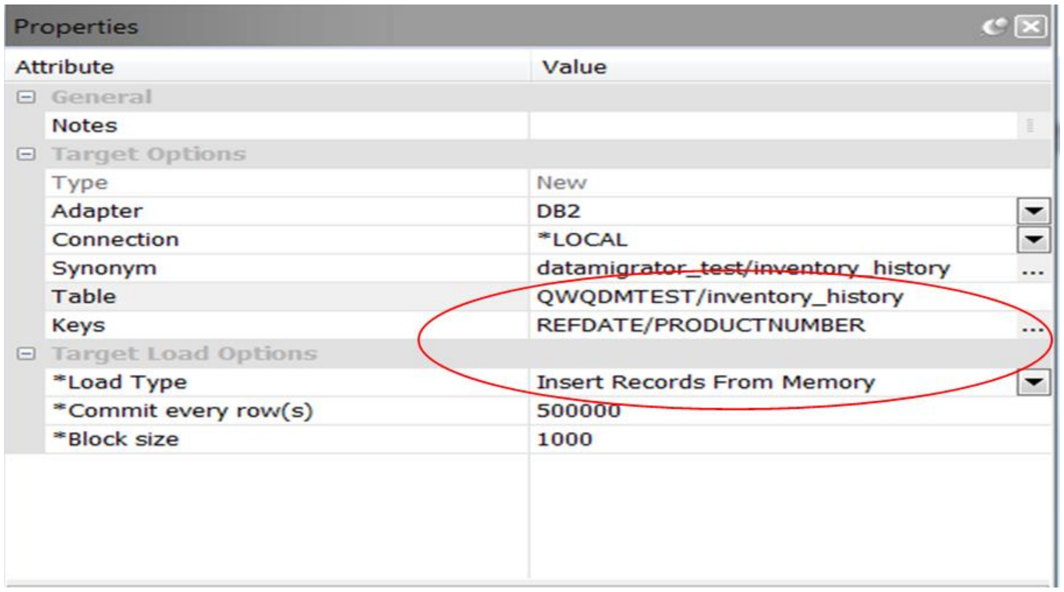
Available Field(s):

Name	Table	F
REFDATE	(INVENTORY_HISTORY)	Y
Product,Number	(INVENTORY_HISTORY)	A
Product Type	(INVENTORY_HISTORY)	A
Product Category	(INVENTORY_HISTORY)	A
Product Name	(INVENTORY_HISTORY)	A
Model	(INVENTORY_HISTORY)	A
Quantity,In Stock	(INVENTORY_HISTORY)	I1
Price	(INVENTORY_HISTORY)	P
Cost	(INVENTORY_HISTORY)	P

Selected Field(s):

	Field
1	REFDATE
2	PRODUCTNUMBER

Figure 106: select REFDATE and PRODUCTNUMBER as Keys

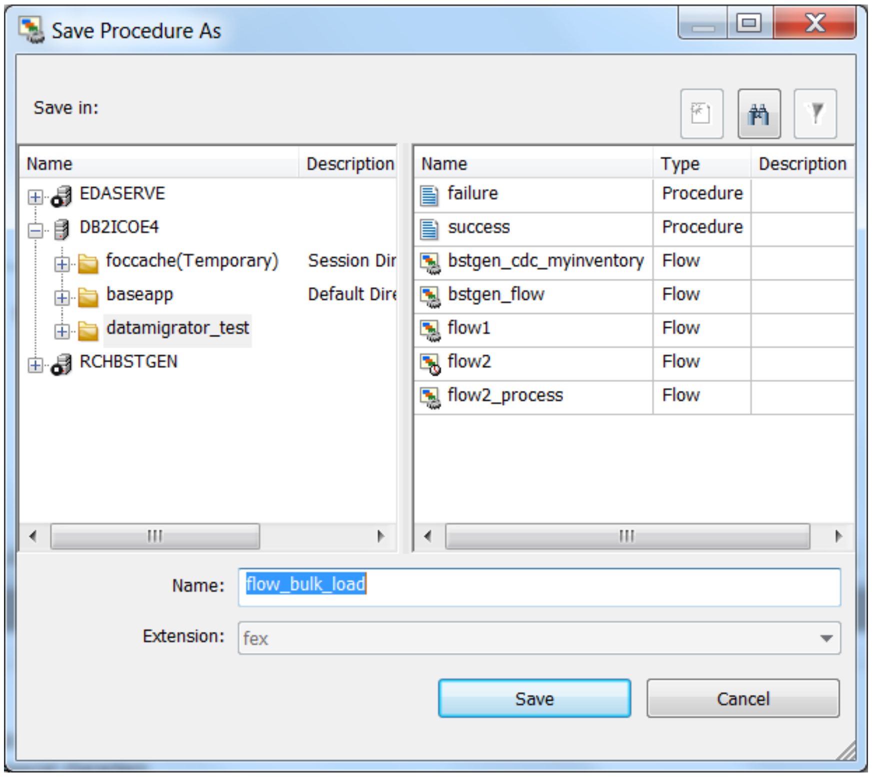


Properties

Attribute	Value
General	
Notes	
Target Options	
Type	New
Adapter	DB2
Connection	*LOCAL
Synonym	datamigrator_test/inventory_history
Table	QWQDMTEST/inventory_history
Keys	REFDATE/PRODUCTNUMBER
Target Load Options	
*Load Type	Insert Records From Memory
*Commit every row(s)	500000
*Block size	1000

Figure 107: Data Target properties with the three important attributes set

Save the flow. Do a *Save As* and give it a name of **flow_bulk_load.**



Save Procedure As

Save in:

Name	Description
EDASERVE	
DB2ICOE4	
foccache(Temporary)	Session Dir
baseapp	Default Dire
datamigrator_test	
RCHBSTGEN	

Name	Type	Description
failure	Procedure	
success	Procedure	
bstgen_cdc_myinventory	Flow	
bstgen_flow	Flow	
flow1	Flow	
flow2	Flow	
flow2_process	Flow	

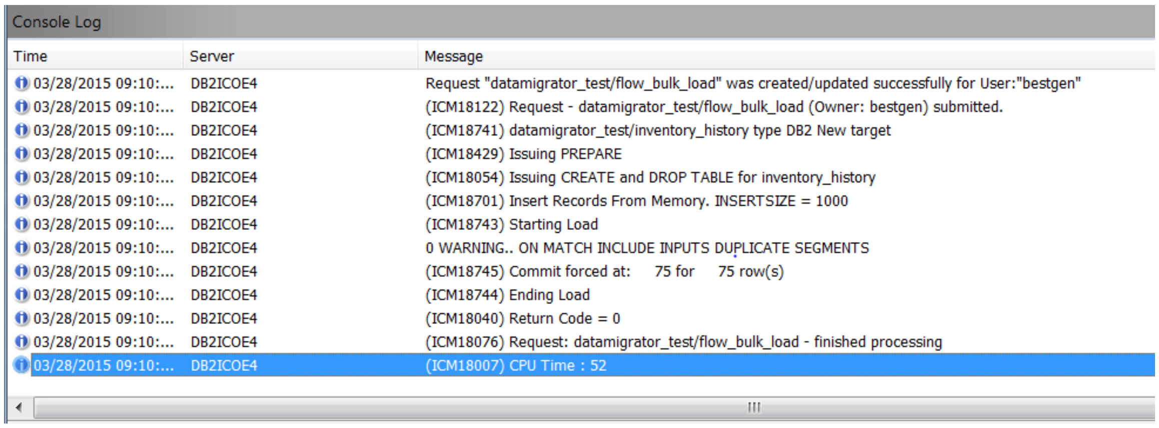
Name: flow_bulk_load

Extension: fex

Figure 108: Save as flow_bulk_load

Select *Save* to save the create flow.

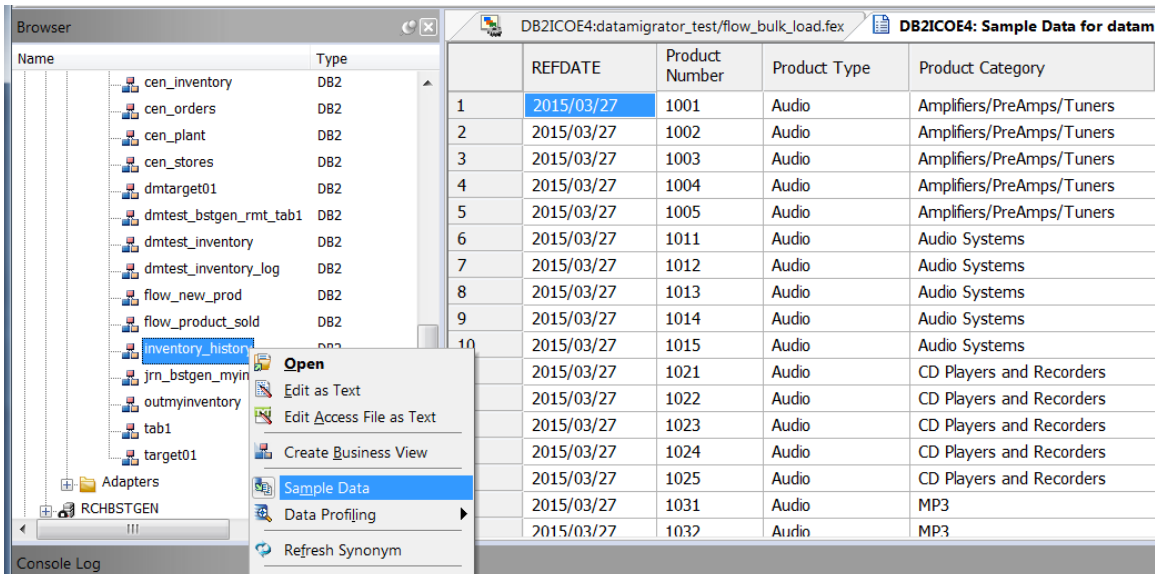
On the left navigation tree, find this new flow, **flow_bulk_load**, right click on it and select *Run*. This will create the target table, **inventory_history**, and populate it with all the records from the data source. In the Console Log we can see the processing that took place when the flow was run.



```
Time                 Server      Message
03/28/2015 09:10:... DB2ICOE4    Request "datamigrator_test/flow_bulk_load" was created/updated successfully for User:"bestgen"
03/28/2015 09:10:... DB2ICOE4    (ICM18122) Request - datamigrator_test/flow_bulk_load (Owner: bestgen) submitted.
03/28/2015 09:10:... DB2ICOE4    (ICM18741) datamigrator_test/inventory_history type DB2 New target
03/28/2015 09:10:... DB2ICOE4    (ICM18429) Issuing PREPARE
03/28/2015 09:10:... DB2ICOE4    (ICM18054) Issuing CREATE and DROP TABLE for inventory_history
03/28/2015 09:10:... DB2ICOE4    (ICM18701) Insert Records From Memory. INSERTSIZE = 1000
03/28/2015 09:10:... DB2ICOE4    (ICM18743) Starting Load
03/28/2015 09:10:... DB2ICOE4    0 WARNING.. ON MATCH INCLUDE INPUTS DUPLICATE SEGMENTS
03/28/2015 09:10:... DB2ICOE4    (ICM18745) Commit forced at:     75 for     75 row(s)
03/28/2015 09:10:... DB2ICOE4    (ICM18744) Ending Load
03/28/2015 09:10:... DB2ICOE4    (ICM18040) Return Code = 0
03/28/2015 09:10:... DB2ICOE4    (ICM18076) Request: datamigrator_test/flow_bulk_load - finished processing
03/28/2015 09:10:... DB2ICOE4    (ICM18007) CPU Time : 52
```

Figure 109: Console Log showing flow_bulk_load flow creating and populating inventory_history with 75 rows

Refresh the left navigation tree by right clicking on the server name (your system name) near the top of the tree and selecting *Refresh*. Now look under the datamigrator_test folder and you should see synonym inventory_history has shown up. Right click on the synonym and select Sample Data.



Name	Type
cen_inventory	DB2
cen_orders	DB2
cen_plant	DB2
cen_stores	DB2
dmtarget01	DB2
dmtest_bstgen_rmt_tab1	DB2
dmtest_inventory	DB2
dmtest_inventory_log	DB2
flow_new_prod	DB2
flow_product_sold	DB2
inventory_history	DB2
jrn_bstgen_myin	
outmyinventory	
tab1	
target01	
Adapters	
RCHBSTGEN	

	REFDATE	Product Number	Product Type	Product Category
1	2015/03/27	1001	Audio	Amplifiers/PreAmps/Tuners
2	2015/03/27	1002	Audio	Amplifiers/PreAmps/Tuners
3	2015/03/27	1003	Audio	Amplifiers/PreAmps/Tuners
4	2015/03/27	1004	Audio	Amplifiers/PreAmps/Tuners
5	2015/03/27	1005	Audio	Amplifiers/PreAmps/Tuners
6	2015/03/27	1011	Audio	Audio Systems
7	2015/03/27	1012	Audio	Audio Systems
8	2015/03/27	1013	Audio	Audio Systems
9	2015/03/27	1014	Audio	Audio Systems
10	2015/03/27	1015	Audio	Audio Systems
	2015/03/27	1021	Audio	CD Players and Recorders
	2015/03/27	1022	Audio	CD Players and Recorders
	2015/03/27	1023	Audio	CD Players and Recorders
	2015/03/27	1024	Audio	CD Players and Recorders
	2015/03/27	1025	Audio	CD Players and Recorders
	2015/03/27	1031	Audio	MP3
	2015/03/27	1032	Audio	MP3

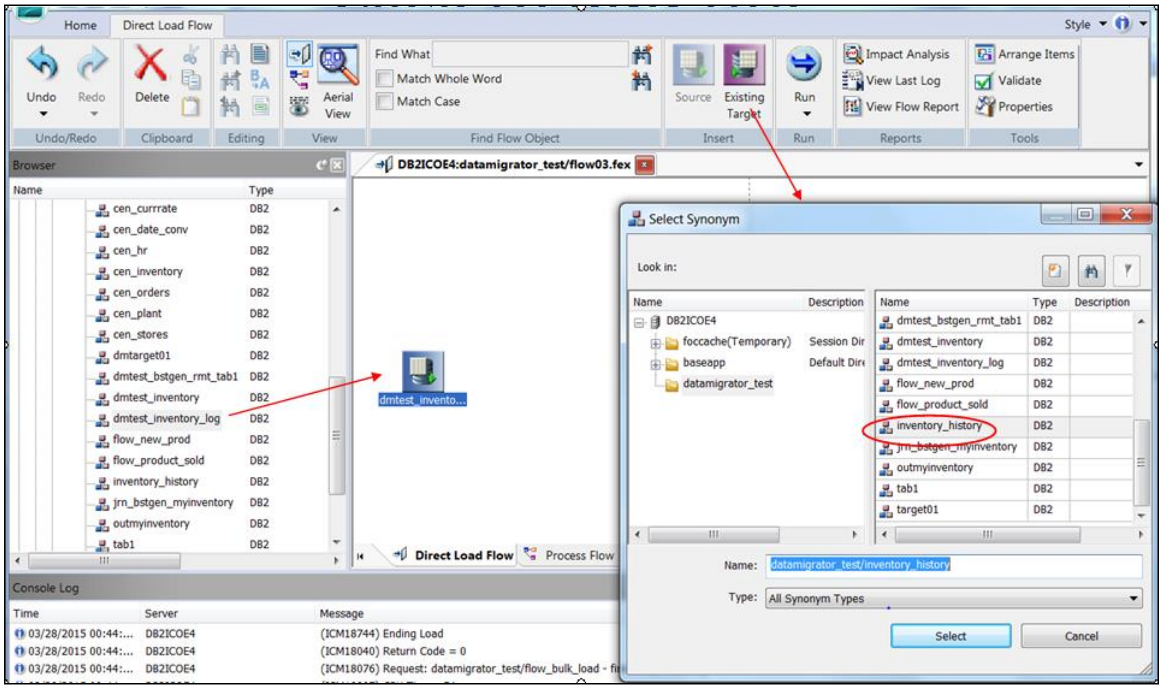
Figure 110: Sample Data against the newly create inventory_history

Close both the sample data tab and the flow_bulk_load tab on the palette.

Creating CDC Maintenance Flow

Now we're ready to create the second flow. This flow will capture any changes in the data source table and apply them to the target table inventory_history. For this project we want to have inventory_history contain a history of our inventory as it changes over time. That way we can go back and see how our inventory fluctuates. We will only capture inventory when it changes.

To build the flow, right click on the datamigrator_test folder and specify *New → Direct Load Flow*. A Direct Load Flow implies the data will be coming in and being processed right away. Drag in **dmtest_inventory_log** from the left navigation tree onto the left side of the flow palette. Next, double click on *Existing Target* from the ribbon and select **inventory_history**.



Name	Type
cen_currrate	DB2
cen_date_conv	DB2
cen_hr	DB2
cen_inventory	DB2
cen_orders	DB2
cen_plant	DB2
cen_stores	DB2
dmtarget01	DB2
dmtest_bstgen_rmt_tab1	DB2
dmtest_inventory	DB2
dmtest_inventory_log	DB2
flow_new_prod	DB2
flow_product_sold	DB2
inventory_history	DB2
jrn_bstgen_myinventory	DB2
outmyinventory	DB2
tab1	DB2

Name	Description
DB2ICOE4	
foccache(Temporary)	Session Dir
baseapp	Default Dire
datamigrator_test	

Name	Type	Description
dmtest_bstgen_rmt_tab1	DB2	
dmtest_inventory	DB2	
dmtest_inventory_log	DB2	
flow_new_prod	DB2	
flow_product_sold	DB2	
inventory_history	DB2	
jrn_bstgen_myinventory	DB2	
outmyinventory	DB2	
tab1	DB2	
target01	DB2	

Name: datamigrator_test/inventory_history

Type: All Synonym Types

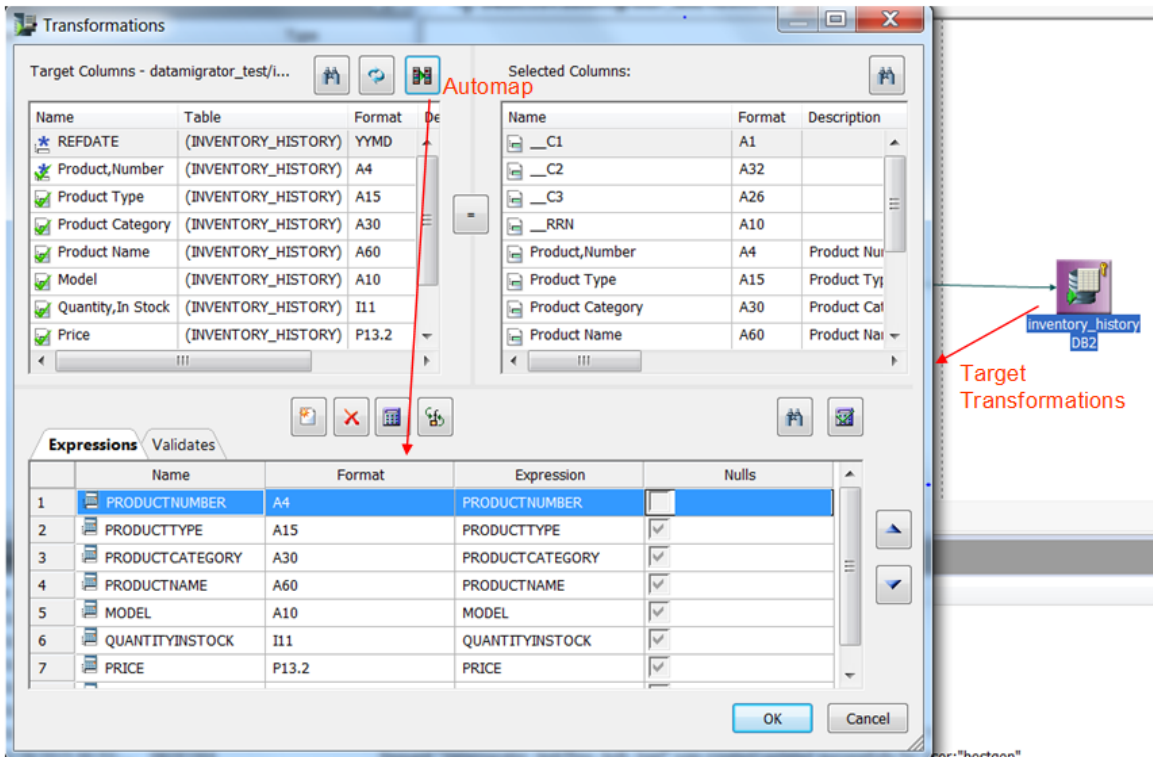
Console Log

```
Time                 Server     Message
03/28/2015 00:44:... DB2ICOE4   (ICM18744) Ending Load
03/28/2015 00:44:... DB2ICOE4   (ICM18040) Return Code = 0
03/28/2015 00:44:... DB2ICOE4   (ICM18076) Request: datamigrator_test/flow_bulk_load - fi
```

Figure 111: Drag in log, double click Existing Target and select inventory_history

Click *Select*.

The **inventory_history** synonym should now show up on the right of the palette and be connected to the **dmtest_inventory_log** synonym. Right click on inventory_history on the palette and choose *Target Transformations*. In the Transformations window, click the automap button at the top to have all matching columns automatically selected.



Target Columns - datamigrator_test/i...

Name	Table	Format	De
REFDATE	(INVENTORY_HISTORY)	YYMD	
Product,Number	(INVENTORY_HISTORY)	A4	
Product Type	(INVENTORY_HISTORY)	A15	
Product Category	(INVENTORY_HISTORY)	A30	
Product Name	(INVENTORY_HISTORY)	A60	
Model	(INVENTORY_HISTORY)	A10	
Quantity,In Stock	(INVENTORY_HISTORY)	I11	
Price	(INVENTORY_HISTORY)	P13.2	

Selected Columns:

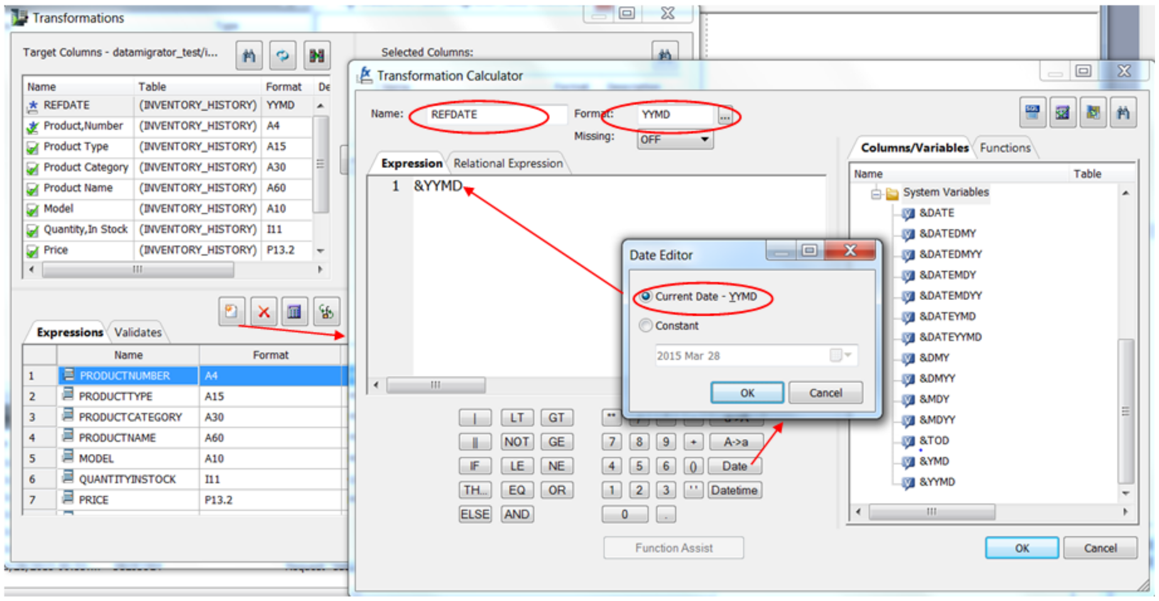
Name	Format	Description
__C1	A1	
__C2	A32	
__C3	A26	
__RRN	A10	
Product,Number	A4	Product Nu
Product Type	A15	Product Ty
Product Category	A30	Product Ca
Product Name	A60	Product Na

	Name	Format	Expression	Nulls
1	PRODUCTNUMBER	A4	PRODUCTNUMBER	☐
2	PRODUCTTYPE	A15	PRODUCTTYPE	✓
3	PRODUCTCATEGORY	A30	PRODUCTCATEGORY	✓
4	PRODUCTNAME	A60	PRODUCTNAME	✓
5	MODEL	A10	MODEL	✓
6	QUANTITYINSTOCK	I11	QUANTITYINSTOCK	✓
7	PRICE	P13.2	PRICE	✓

Figure 112: Target Transformations, Automap

Now we want to add a current date field like we did for bulk load. Select the *Insert Transforms* button. In the Transformation Calculator:

- specify a name **REFDATE**.
- choose the *Date* button. Select *Current Date – YYMD*.
- For the Format specify **YYMD**.



Transformations

Target Columns - datamigrator_test/i...

Name	Table	Format	De
REFDATE	(INVENTORY_HISTORY)	YYMD	
Product,Number	(INVENTORY_HISTORY)	A4	
Product Type	(INVENTORY_HISTORY)	A15	
Product Category	(INVENTORY_HISTORY)	A30	
Product Name	(INVENTORY_HISTORY)	A60	
Model	(INVENTORY_HISTORY)	A10	
Quantity,In Stock	(INVENTORY_HISTORY)	I11	
Price	(INVENTORY_HISTORY)	P13.2	

Expressions | Validates

	Name	Format
1	PRODUCTNUMBER	A4
2	PRODUCTTYPE	A15
3	PRODUCTCATEGORY	A30
4	PRODUCTNAME	A60
5	MODEL	A10
6	QUANTITYINSTOCK	I11
7	PRICE	P13.2

Selected Columns:

Transformation Calculator

Name: REFDATE Format: YYMD Missing: OFF

Expression | Relational Expression

```
1  &YYMD
```

Date Editor

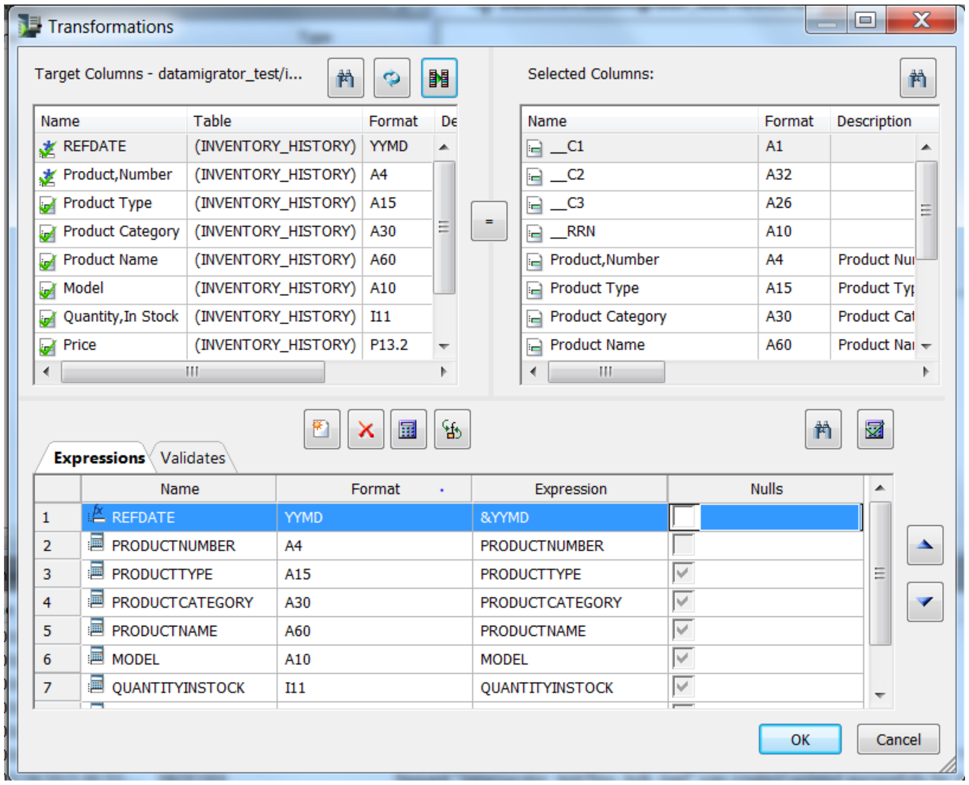
(● Current Date - YYMD) ○ Constant 2015 Mar 28

Columns/Variables | Functions

Name	Table
System Variables	
&DATE	
&DATEDMY	
&DATEDMYY	
&DATEMDY	
&DATEMDYY	
&DATEYMD	
&DATEYYMD	
&DMY	
&DMYY	
&MDY	
&MDYY	
&TOD	
&YMD	
&YYMD	

Figure 113: Insert Transform for Current Date

Select OK to create the new **REFDATE** column. Now move REFDATE to the top of the column list by highlighting it and using the up arrow key to move it to the top.



Transformations

Target Columns - datamigrator_test/i...

Name	Table	Format	De
REFDATE	(INVENTORY_HISTORY)	YYMD	
Product,Number	(INVENTORY_HISTORY)	A4	
Product Type	(INVENTORY_HISTORY)	A15	
Product Category	(INVENTORY_HISTORY)	A30	
Product Name	(INVENTORY_HISTORY)	A60	
Model	(INVENTORY_HISTORY)	A10	
Quantity,In Stock	(INVENTORY_HISTORY)	I11	
Price	(INVENTORY_HISTORY)	P13.2	

Selected Columns:

Name	Format	Description
__C1	A1	
__C2	A32	
__C3	A26	
__RRN	A10	
Product,Number	A4	Product Nu
Product Type	A15	Product Ty
Product Category	A30	Product Ca
Product Name	A60	Product Na

Expressions | Validates

	Name	Format	Expression	Nulls
1	REFDATE	YYMD	&YYMD	☐
2	PRODUCTNUMBER	A4	PRODUCTNUMBER	☐
3	PRODUCTTYPE	A15	PRODUCTTYPE	✓
4	PRODUCTCATEGORY	A30	PRODUCTCATEGORY	✓
5	PRODUCTNAME	A60	PRODUCTNAME	✓
6	MODEL	A10	MODEL	✓
7	QUANTITYINSTOCK	I11	QUANTITYINSTOCK	✓

Figure 114: REFDATE moved to the top of the list

Select OK.

Back on the palette, right click on the inventory_history icon and select *Properties*.

This flow will be capturing changes to the inventory table (via the journal) and putting it into the inventory_history table. We will be capturing the change once a day. However, for any given day it is certainly possible that there could be multiple changes to a particular product in the inventory. Which of these changes do we capture in the history table for that day?

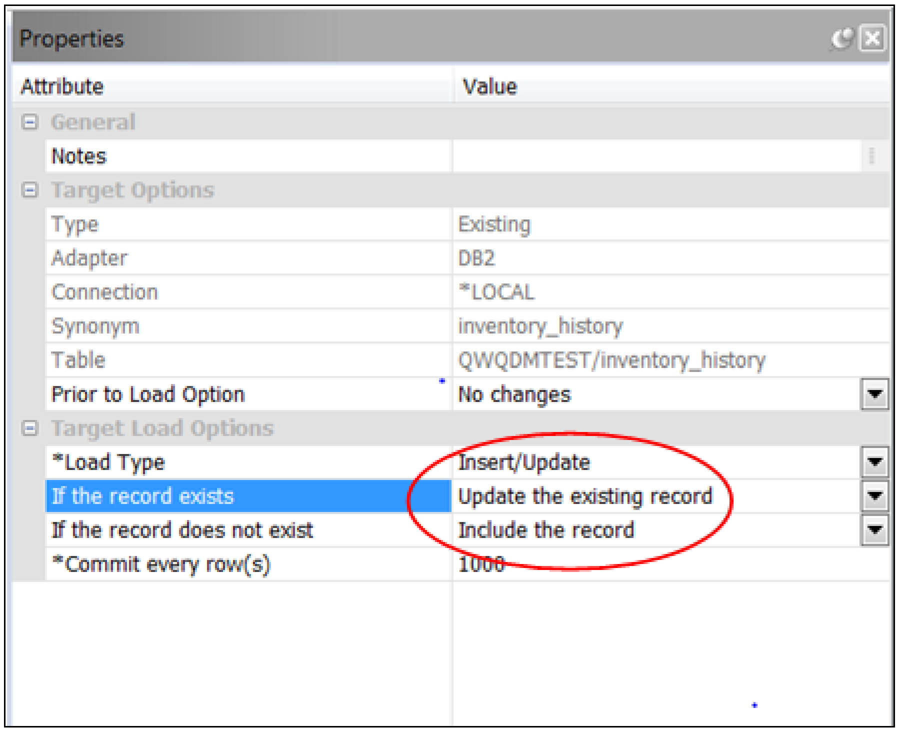
Since we will only keep one inventory change per day per product, let's decide to keep the last change for a given day. That means two things:

- For a row read from the journal, if we see there is already a row in the inventory_history table for the given product on the current date, we will replace that row in the history table with the incoming row. This means the last product inventory change for the current date will ultimately be the one we retain in the history table.
- For a row read from the journal, if there isn't a row in the history table for the product for the current date, add it to the history table.

We prepared for this by defining the *Key* for the history table to be REFDATE and PRODUCTNUMBER. This key defines what it means to have a 'matching' row in the history table. Therefore, anytime a new row coming in from the journal has the same REFDATE and PRODUCTNUMBER value as an existing row in the inventory_history table, we have a match and must make a decision what to do. In this case we want to replace the row in the history table with the incoming row.

To implement these rules, set the following values in the properties:

- *Load Type* set to *Insert/Update*.
- For *If the record exists* choose *Update the existing record*. This will cause the incoming record to be used, meaning it will overwrite the existing record in the history table.
- For *If the record does not exist* choose *Include the record*. This will cause the incoming record to be added to the history table if there is not an entry for the product on the current date.



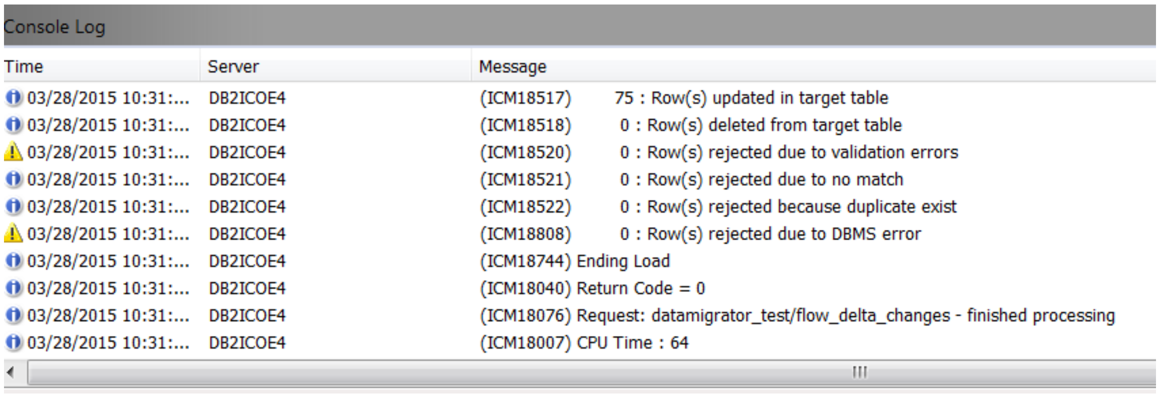
Properties

Attribute	Value
General	
Notes	
Target Options	
Type	Existing
Adapter	DB2
Connection	*LOCAL
Synonym	inventory_history
Table	QWQDMTEST/inventory_history
Prior to Load Option	No changes
Target Load Options	
*Load Type	Insert/Update
If the record exists	Update the existing record
If the record does not exist	Include the record
*Commit every row(s)	1000

Figure 115: inventory_history properties with attributes set

Save this flow using *Save As*. Give it a name of **flow_delta_changes.**

Let's run this Direct Load Flow. Find **flow_delta_changes** in the left navigation tree, right click on it, and select *Run*. Look at the Console log at the bottom of DMC.



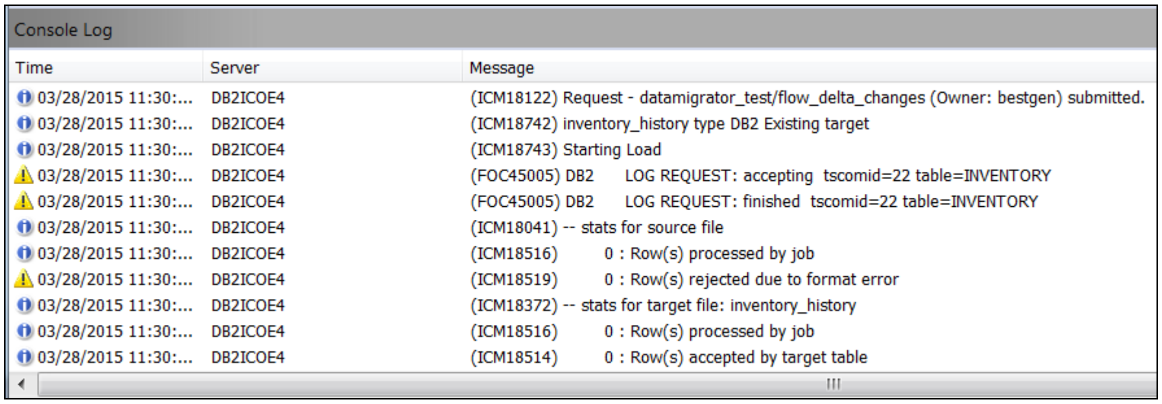
Console Log

Time	Server	Message
03/28/2015 10:31:...	DB2ICOE4	(ICM18517) 75 : Row(s) updated in target table
03/28/2015 10:31:...	DB2ICOE4	(ICM18518) 0 : Row(s) deleted from target table
03/28/2015 10:31:...	DB2ICOE4	(ICM18520) 0 : Row(s) rejected due to validation errors
03/28/2015 10:31:...	DB2ICOE4	(ICM18521) 0 : Row(s) rejected due to no match
03/28/2015 10:31:...	DB2ICOE4	(ICM18522) 0 : Row(s) rejected because duplicate exist
03/28/2015 10:31:...	DB2ICOE4	(ICM18808) 0 : Row(s) rejected due to DBMS error
03/28/2015 10:31:...	DB2ICOE4	(ICM18744) Ending Load
03/28/2015 10:31:...	DB2ICOE4	(ICM18040) Return Code = 0
03/28/2015 10:31:...	DB2ICOE4	(ICM18076) Request: datamigrator_test/flow_delta_changes - finished processing
03/28/2015 10:31:...	DB2ICOE4	(ICM18007) CPU Time : 64

Figure 116: Console log from first flow_delta_changes run

Notice that it processed 75 rows. In other words, it reprocessed the 75 rows that the bulk load already processed. This is because the journal contains all the entries in inventory, since we created/populated the table under commit against this journal. If this had been case where inventory was an older file with records existing prior to journaling being started, the bulk load would capture all the base records, thereby providing the base records for the subsequent direct flow runs.

Let’s run the Direct Load Flow a second time. Right click on **flow_delta_changes** in the tree and select *Run*. Look at the Console log at the bottom of DMC this time.



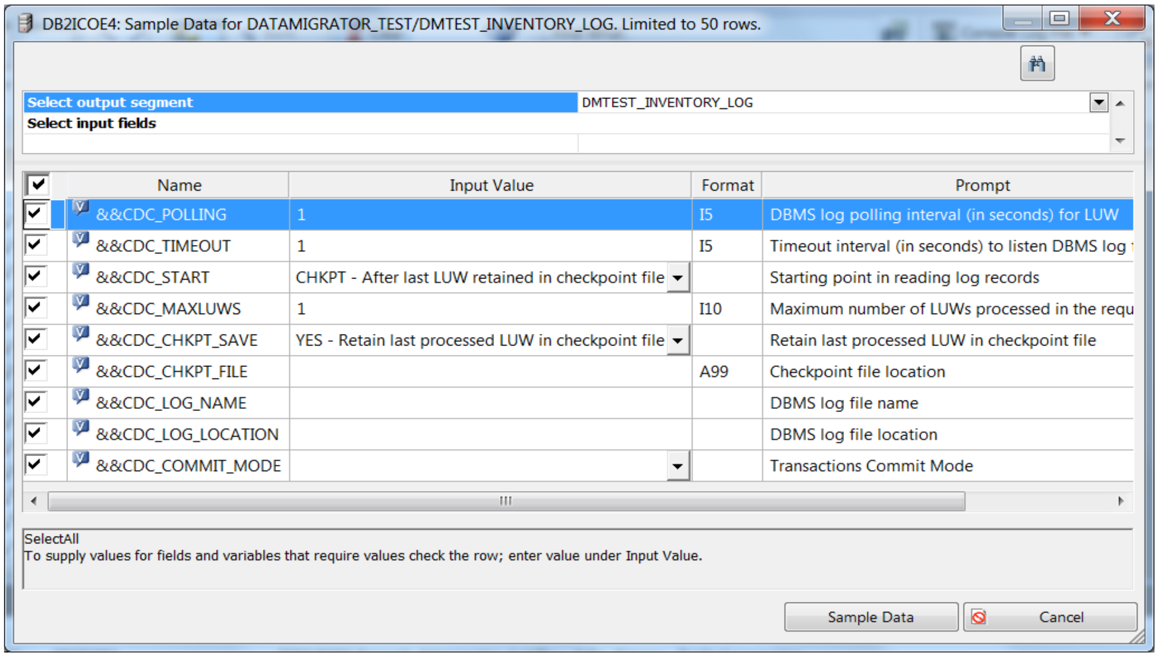
Console Log

```
Time                 Server     Message
03/28/2015 11:30:... DB2ICOE4   (ICM18122) Request - datamigrator_test/flow_delta_changes (Owner: bestgen) submitted.
03/28/2015 11:30:... DB2ICOE4   (ICM18742) inventory_history type DB2 Existing target
03/28/2015 11:30:... DB2ICOE4   (ICM18743) Starting Load
03/28/2015 11:30:... DB2ICOE4   (FOC45005) DB2      LOG REQUEST: accepting  tscomid=22 table=INVENTORY
03/28/2015 11:30:... DB2ICOE4   (FOC45005) DB2      LOG REQUEST: finished  tscomid=22 table=INVENTORY
03/28/2015 11:30:... DB2ICOE4   (ICM18041) -- stats for source file
03/28/2015 11:30:... DB2ICOE4   (ICM18516)        0 : Row(s) processed by job
03/28/2015 11:30:... DB2ICOE4   (ICM18519)        0 : Row(s) rejected due to format error
03/28/2015 11:30:... DB2ICOE4   (ICM18372) -- stats for target file: inventory_history
03/28/2015 11:30:... DB2ICOE4   (ICM18516)        0 : Row(s) processed by job
03/28/2015 11:30:... DB2ICOE4   (ICM18514)        0 : Row(s) accepted by target table
```

Figure 117: Console log second flow_delta_changes run

On this second run, no rows were processed. This shows an important aspect of table log synonyms called a *log checkpoint file*. Each time a flow involving a table log synonym runs, a checkpoint file (a file in IFS) is updated with information about the last journal entry read. By default when the flow is run again, it starts reading in the journal after the last position recorded in the checkpoint file. If subsequent journal entries are add, the flow will pick up those new journal entries and apply them and then once again update the checkpoint file to indicate a new ending position in the journal. In this way we can run the flow_delta_changes flow over and over, day after day, confident it will only process new entries.

Now that we have seen the checkpoint process in action, let’s step back and take a closer look at it. In the left navigation tree, right click on the table log synonym **dmtest_inventory_log** and select *Sample Data*. Check box all the &&CDC_ variables and select *Sample Data*.


DB2ICOE4: Sample Data for DATAMIGRATOR_TEST/DMTEST_INVENTORY_LOG. Limited to 50 rows.

Select output segment	DMTEST_INVENTORY_LOG
Select input fields	

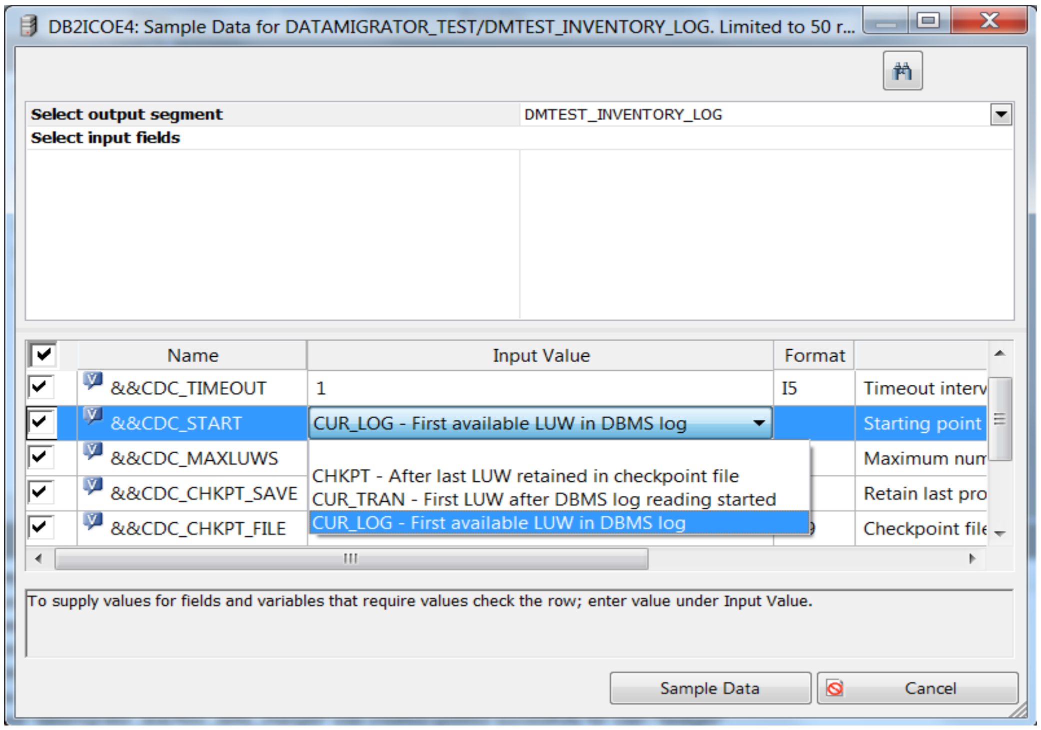
✓	Name	Input Value	Format	Prompt
✓	&&CDC_POLLING	1	I5	DBMS log polling interval (in seconds) for LUW
✓	&&CDC_TIMEOUT	1	I5	Timeout interval (in seconds) to listen DBMS log
✓	&&CDC_START	CHKPT - After last LUW retained in checkpoint file		Starting point in reading log records
✓	&&CDC_MAXLUWS	1	I10	Maximum number of LUWs processed in the requ
✓	&&CDC_CHKPT_SAVE	YES - Retain last processed LUW in checkpoint file		Retain last processed LUW in checkpoint file
✓	&&CDC_CHKPT_FILE		A99	Checkpoint file location
✓	&&CDC_LOG_NAME			DBMS log file name
✓	&&CDC_LOG_LOCATION			DBMS log file location
✓	&&CDC_COMMIT_MODE			Transactions Commit Mode

SelectAll
To supply values for fields and variables that require values check the row; enter value under Input Value.

Figure 118: Check all parameter on Sample Data

Notice that no records were displayed. This is because the checkpoint file has saved the last read position, which was the last entry in the journal.

Now let’s sample data again but this time with one difference. In the left navigation tree, right click on the table log synonym **dmtest_inventory_log** again and select *Sample Data*. Check box all the &&CDC_ variables as before. But this time click on the dropdown box for *&&CDC_START* and choose *CUR_LOG – First available LUW in DBMS log*.


DB2ICOE4: Sample Data for DATAMIGRATOR_TEST/DMTEST_INVENTORY_LOG. Limited to 50 r...

Select output segment	DMTEST_INVENTORY_LOG
Select input fields	

✓	Name	Input Value	Format	
✓	&&CDC_TIMEOUT	1	I5	Timeout interv
✓	&&CDC_START	CUR_LOG - First available LUW in DBMS log		Starting point
✓	&&CDC_MAXLUWS			Maximum num
✓	&&CDC_CHKPT_SAVE			Retain last pro
✓	&&CDC_CHKPT_FILE			Checkpoint file

CHKPT - After last LUW retained in checkpoint file
CUR_TRAN - First LUW after DBMS log reading started
CUR_LOG - First available LUW in DBMS log

To supply values for fields and variables that require values check the row; enter value under Input Value.

Figure 119: Change &&CDC_START to be the beginning of the log (journal)

Now select *Sample Data*.

This time we got a full sample of records from the log (journal). This shows a very useful feature for CDC processing. It is possible to control whether to: 1. maintain position in the journal as entries are processed (the default), 2. restart from the beginning of the journal, or 3. start from a point in time in the journal that corresponds to when the flow job started.

> **Note**: The change of value for the *&&CDC_START* variable was temporary, just for our sample data. It did not change the value permanently for the synonym. To do a permanent change, we would have to open the synonym, select the variable, change its default, then save the synonym.

Two other variables that work in conjunction with *&&CDC_START* are *&&CDC_CHKPT_SAVE* and *&&CDC_CHKPT_FILE*.

- &&CDC_CHKPT_SAVE defines whether the checkpoint file is updated at the end of the journal read processing in the flow. Normally this is set to YES to indicate the position should be saved (updated). However, in test scenarios it can be useful to not save position so that repeatedly running the flow will process the same entries over and over.
- &&CDC_CHKPT_FILE defines the (IFS) file that will retain the checkpoint information. By default the directory and name is

/qibm/userdata/qwebqry/apps/<*application directory*>/<*synonymname*>.chp. However, you can change this to a different name if you wish. This is sometimes useful if you want to maintain different checkpoint positions for different flows.

OK, now let's see the delta flow in action. First we need to make some row changes in the data source so that new journal entries are created. Through an SQL interface, update a few rows in the source table with the following SQL statements. Note: make sure to run under commitment control like we did before:

```
UPDATE QWQDMTEST.INVENTORY
SET QUANTITYINSTOCK = QUANTITYINSTOCK - 150
WHERE PRODUCTNUMBER BETWEEN '1000' and '1010';

COMMIT;
```

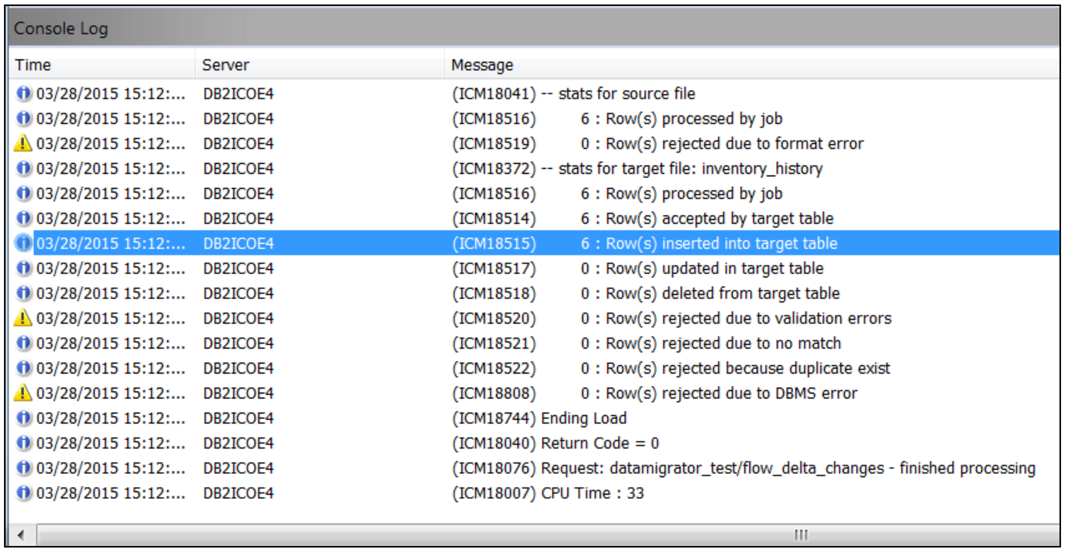
This will reduce the inventory of all products within the productnumber range by 150.

Ideally we could just run the delta flow and see the changes apply. However, remember we are using the current date as the value for CURDATE. Assuming you are running these flows on the same day, our inventory_history file will only have the current date as the value of the CURDATE column in all its rows. We would like to see our UPDATE above show as a history i.e. over more than one day. Therefore, let's cheat and first update the existing rows of target table inventory_history to set the date to an older date. Run this SQL statement:

```
UPDATE QWQDMTEST.INVENTORY_HISTORY
SET REFDATE = REFDATE - 1 DAY;

COMMIT;
```

In effect we change the history table to look like it was populated yesterday.

Now run the Direct Load Flow a third time. Right click on **flow_delta_changes** in the tree and select *Run*. Look at the Console log at the bottom of DMC.

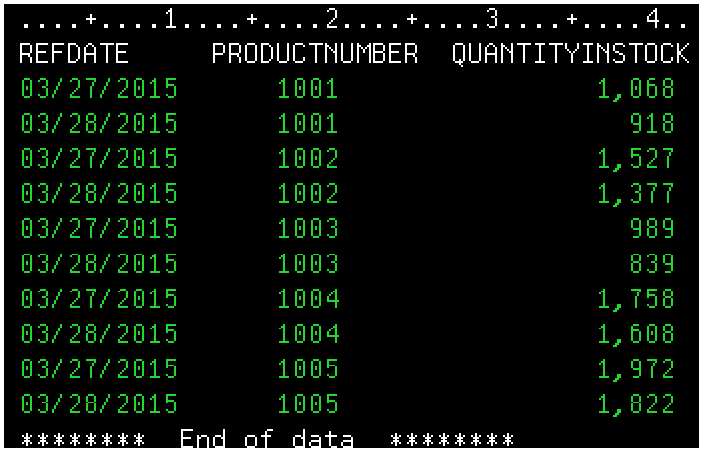


```
Console Log
Time                  Server     Message
03/28/2015 15:12:...  DB2ICOE4   (ICM18041) -- stats for source file
03/28/2015 15:12:...  DB2ICOE4   (ICM18516)       6 : Row(s) processed by job
03/28/2015 15:12:...  DB2ICOE4   (ICM18519)       0 : Row(s) rejected due to format error
03/28/2015 15:12:...  DB2ICOE4   (ICM18372) -- stats for target file: inventory_history
03/28/2015 15:12:...  DB2ICOE4   (ICM18516)       6 : Row(s) processed by job
03/28/2015 15:12:...  DB2ICOE4   (ICM18514)       6 : Row(s) accepted by target table
03/28/2015 15:12:...  DB2ICOE4   (ICM18515)       6 : Row(s) inserted into target table
03/28/2015 15:12:...  DB2ICOE4   (ICM18517)       0 : Row(s) updated in target table
03/28/2015 15:12:...  DB2ICOE4   (ICM18518)       0 : Row(s) deleted from target table
03/28/2015 15:12:...  DB2ICOE4   (ICM18520)       0 : Row(s) rejected due to validation errors
03/28/2015 15:12:...  DB2ICOE4   (ICM18521)       0 : Row(s) rejected due to no match
03/28/2015 15:12:...  DB2ICOE4   (ICM18522)       0 : Row(s) rejected because duplicate exist
03/28/2015 15:12:...  DB2ICOE4   (ICM18808)       0 : Row(s) rejected due to DBMS error
03/28/2015 15:12:...  DB2ICOE4   (ICM18744) Ending Load
03/28/2015 15:12:...  DB2ICOE4   (ICM18040) Return Code = 0
03/28/2015 15:12:...  DB2ICOE4   (ICM18076) Request: datamigrator_test/flow_delta_changes - finished processing
03/28/2015 15:12:...  DB2ICOE4   (ICM18007) CPU Time : 33
```

Figure 120: Console log flow_delta_changes run after updates

On this run, the source table rows we updated were processed. We can verify this by looking into the inventory_history table for those rows using the SQL statement:

```
SELECT REFDATE, PRODUCTNUMBER, QUANTITYINSTOCK
FROM QWQDMTEST.INVENTORY_HISTORY
WHERE PRODUCTNUMBER BETWEEN '1000' and '1010'
ORDER BY PRODUCTNUMBER
```



```
....+....1....+....2....+....3....+....4..
REFDATE       PRODUCTNUMBER  QUANTITYINSTOCK
03/27/2015        1001                1,068
03/28/2015        1001                  918
03/27/2015        1002                1,527
03/28/2015        1002                1,377
03/27/2015        1003                  989
03/28/2015        1003                  839
03/27/2015        1004                1,758
03/28/2015        1004                1,608
03/27/2015        1005                1,972
03/28/2015        1005                1,822
********  End of data  ********
```

We now have two days worth of history for these products. Each day has a different quantityinstock that reflects that we reduced the amount by 150.

If we were planning to actually implement this flow in our environment, the next step would be to schedule the flow_delta_changes flow to run every day to capture any changes from our source environment. However, since we covered scheduling in a prior chapter, we will not cover that here.

Appendix

DataMigrator for i contains many more features than discussed in this user's guide. For more information, check out the help text and online User's Guide. They are available from the main DMC screen by selecting the arrow next to the icon in the upper right.

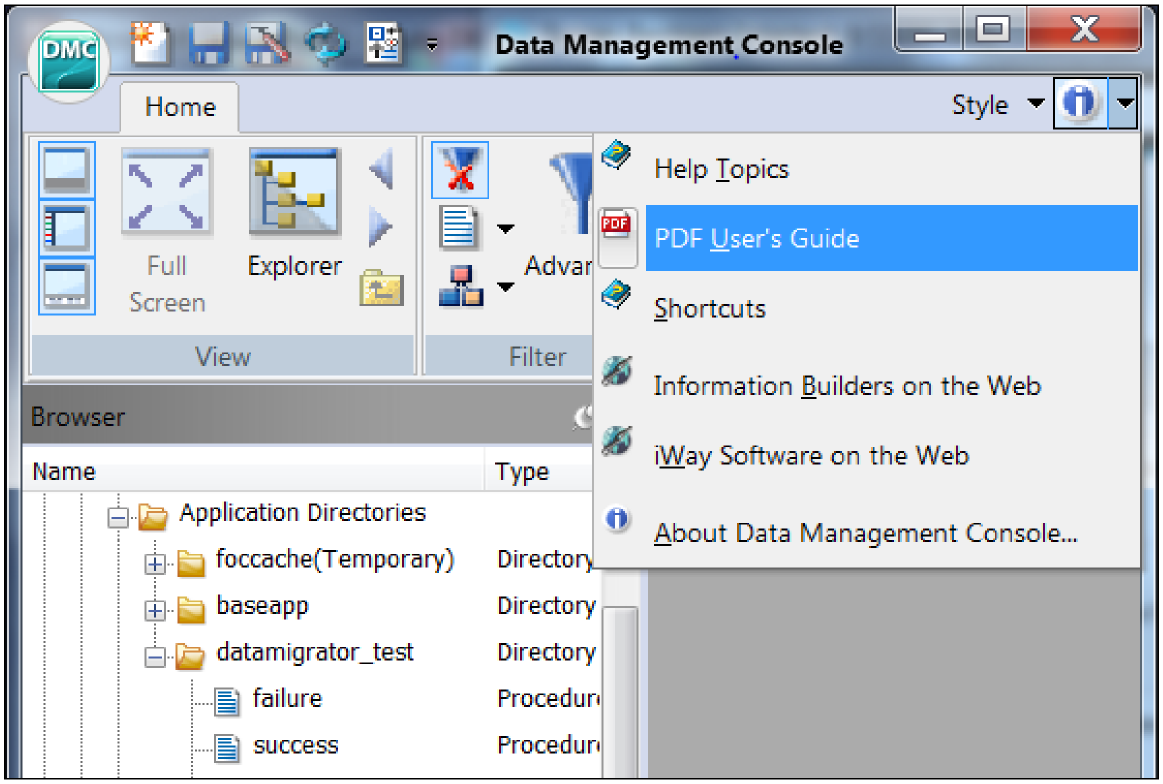


Figure 121: Online Information