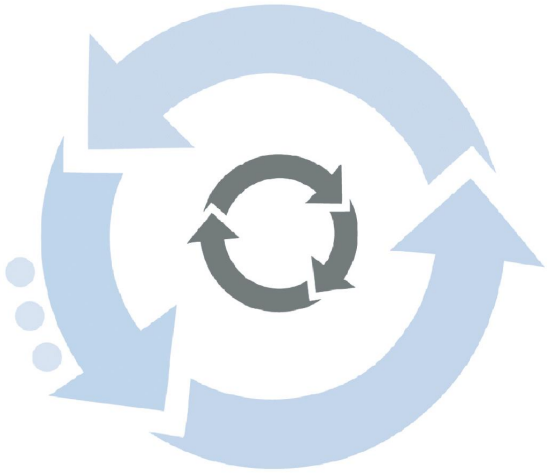
mv.SSIS User Guide



A product from BlueFinity International

BlueFinity

Copyright Notices

Contacting Us

We are always very happy to be able to discuss all aspects of our products with our customers - prospective and current alike. You can contact us via the following means:

Website: www.bluefinity.com
Email: support@bluefinity.com
Address: 10260 SW Greenburg Road, Suite 400, Portland, OR 97223, USA
Address: 575-599 Maxted Road, Hemel Hempstead, Herts, HP2 7DX, UK

Trademark Acknowledgements

Contents

Welcome to mv.SSIS

Firstly, thank you for either purchasing one the mv.SSIS product, or for taking the time to explore the great functionality that it can provide to you and your fellow developers.

This chapter outlines the functionality of the mv.SSIS product.

Introduction

SSIS is a graphical based set of integration tools which are provided with SQL Server. These tools allow for data flow processes to be built in a graphical environment for purposes such as data mining, data migration, reporting and OLAP (Online Analytical Processing).

The BlueFinity SSIS integration component, mv.SSIS, allows data to be read from MultiValue databases into the SSIS environment. This opens up the entire SQL Server set of tools and utilities to MultiValue database users.

SSIS in Brief

SSIS provides a graphical front end to design control flow data processing logic. Once designed, these 'packages' are compiled into '.dtsx' packages which can then be distributed and run on any machine with the SSIS tools installed.

Packages contain two main logic flows, a 'Control Flow' which defines a sequence of logical operations which are processed in sequence. Each step is completed before the next starts e.g.

1. Empty out work tables in a database
2. Populate the work tables with data
3. Perform calculations and update the values in the work table
4. Update OLAP cubes with the data from the work tables
5. Run reports against the OLAP cubes.

This level of control also allows processing loops to be defined e.g.
For each file in a specified folder, read the contents of the file and write it into a specified table.

The second main logic flow is the 'Data Flow'. This allows for the processing of data at the record level. Data is read from a 'Data Source' and passes down a series of 'Data Transformations' to a 'Data Destination'. These transformations can be as simple as changing the data type of fields e.g. varchar(4000) to varchar(2000) or decimal(18,2) to decimal(8,2), or can be more complex like data merges, joins, pivot tables, multicasts etc. Each transformation is represented by an icon in the designer and the icons are linked together to define the logic path.

BlueFinity's SSIS Solution

The mv.SSIS product utilizes BlueFinity's mv.NET product to provide two new objects within the SSIS toolkit:

- An 'mv.NET Connection Manager'
- An 'mv.NET Data Source'

The Connection Manager provides an SSIS-aware connection to an mv.NET-based data provider which then can be used throughout an SSIS package.

Data sources are used within 'Data Flows' and use a connection manager to connect to a data provider and then feed data into the rest of the 'Data Flow'.

To connect to a MultiValue database from within an SSIS package, the user must first add an mv.NET connection manager to the package then within a 'Data Flow Task' add an mv.NET Data Source.

From within the custom SDI data source editor the user can define what data they want to read into the process. The rest of the transformations and destinations are then defined as per the user's requirements using any of the standard SSIS 'Data Flow Transformations' and 'Data Flow Destinations'.

mv.SSIS Prerequisites

This chapter describes the prerequisites necessary for the successful use of mv.SSIS.

mv.NET Dependency

The mv.NET Adapter Objects component set, amongst other things, provides an ADO.NET data provider. It is this ADO.NET data provider that mv.SSIS utilizes in order to retrieve both schema information and data from MultiValue databases.

Therefore, it is important that, before installing and using mv.SSIS, you install either the CID or SRDK products of mv.NET onto the system on which you intend to install and use mv.SSIS.

In order to use Adapter Objects, you will probably need to perform a small amount of preparation work on the MultiValue dictionaries so that it can correctly understand how to move multi-valued data into the strongly typed, 2-dimensional world of SQL. Please refer the mv.NET Adapter Objects Developer Guide for full details on this subject.

SSIS Integration

Although perhaps obvious, it should be stated that in order use mv.SSIS, you must have first installed Microsoft's SSIS product onto the system on which you intend to install and use mv.SSIS. All of mv.SSIS's functionality is accessed from within the SSIS environment.

mv.SSIS in Use

This chapter guides you through the typical use of the mv.SSIS product within the Microsoft SSIS environment.

Introduction

The mv.SSIS product provides a data source into SSIS Data Flow packages for mv.NET compatible databases.

This integration provides two components to the SSIS toolkit:

1. mv.NET Connection Manager
2. mv.NET Data Source

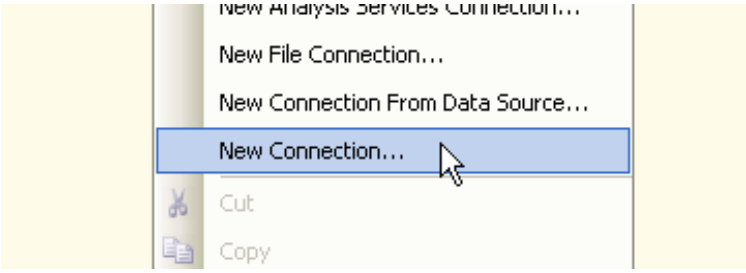
Creating a Simple Data Transformation

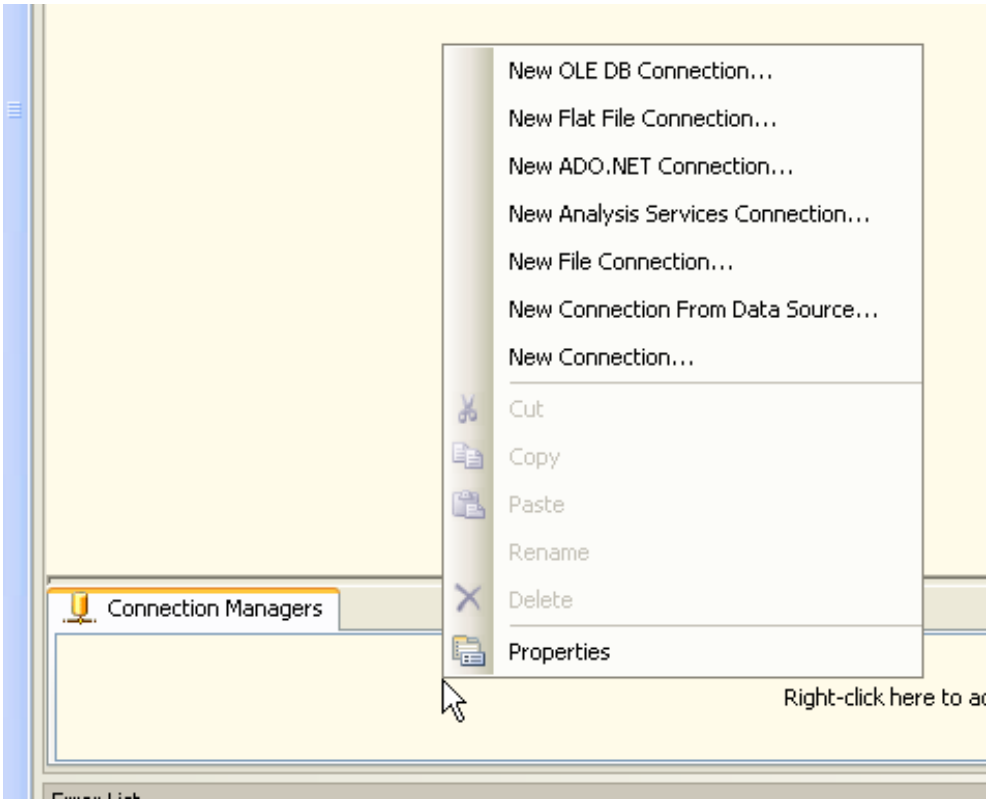
To begin, within the SSIS 'Control Flow' tab, add a 'Data Flow Task' and double click it. This will take you to the data flow designer.

What we now want to do is get access to the multi value database and feed data from it to the main data flow process.

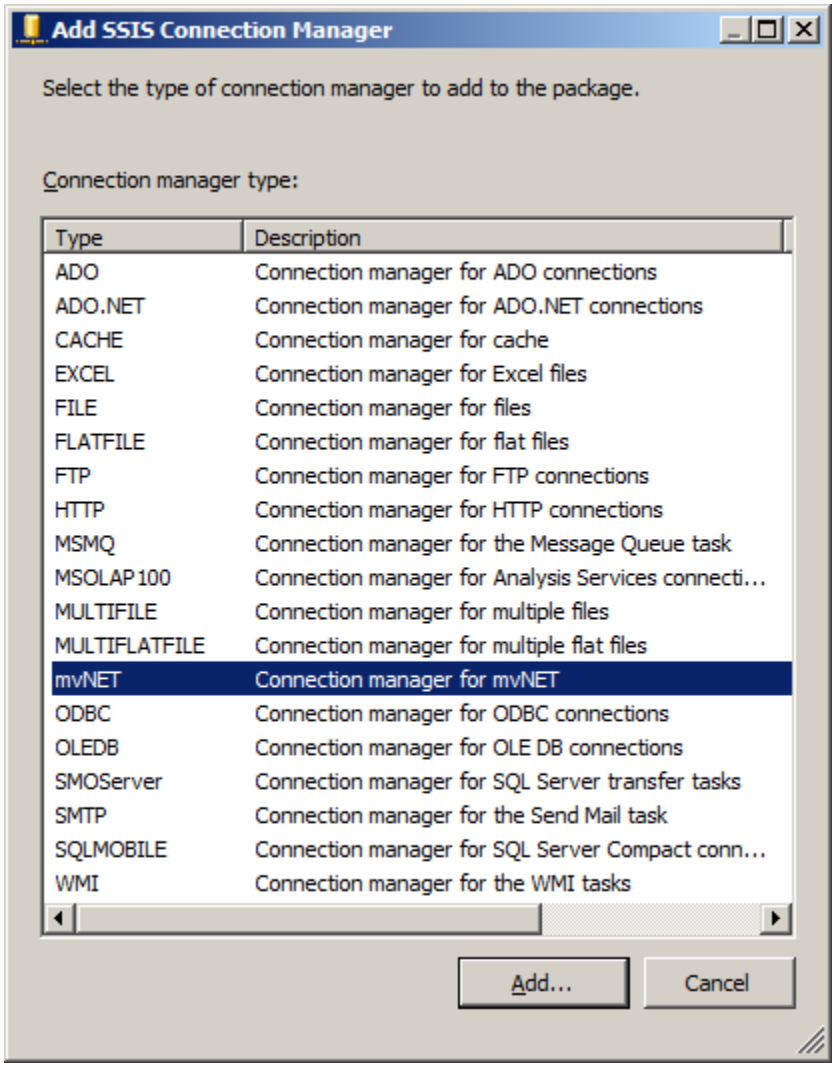
The first step to this is to add a 'Connection Manager' to the package. To do this, right click on the 'Connection Managers' section of the design interface:

and select 'New Connection'

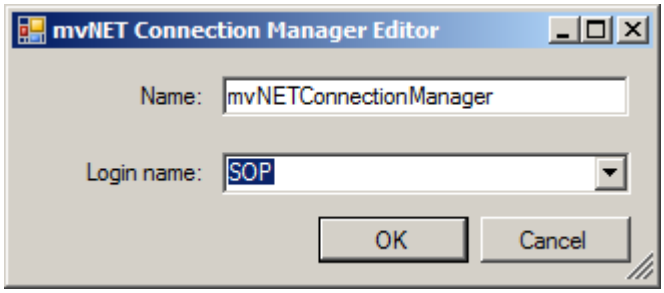




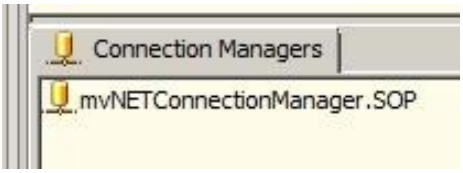
Now select the 'Connection manager for mvNET':



And click the 'Add...' button, this will then display the form below, where you can select the login for the mv.NET database and name the connection:

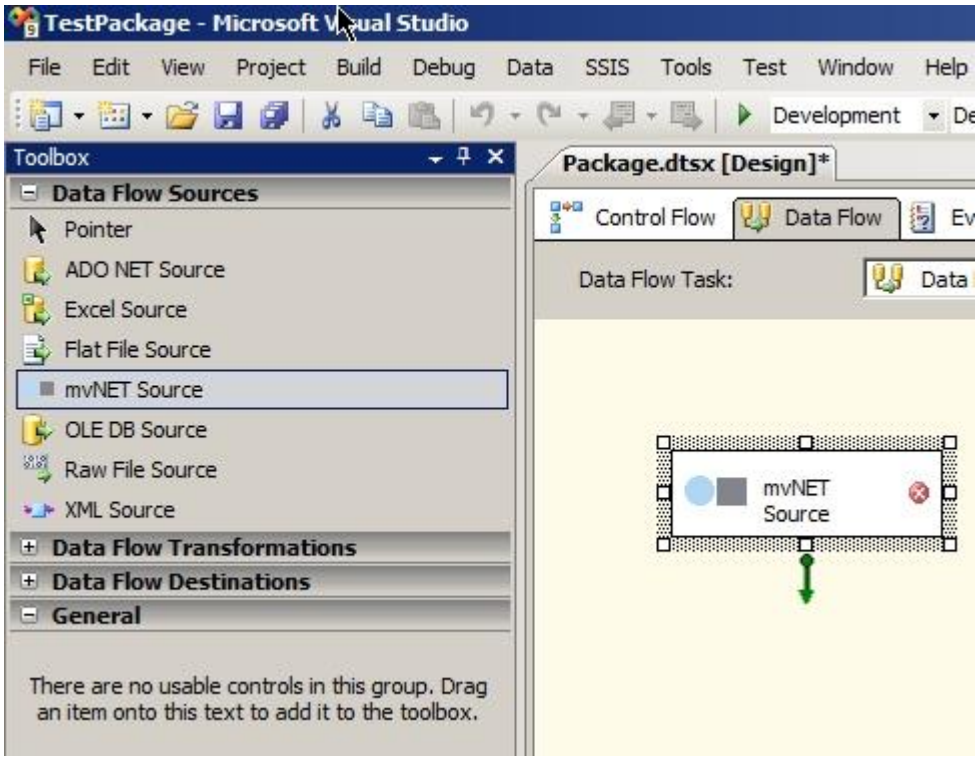


Once you click 'OK' you will have the connection displayed in the designer:

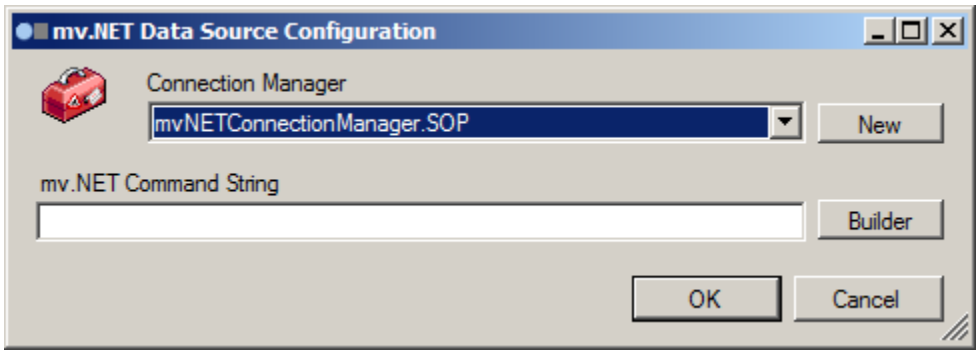


Note: If this is a new project, we must first click in the Data Flow tab and select the option to create a new Data Flow package. Also, the 'mv.NET Source' does not automatically show up in the 'Data Flow Sources' section of the Toolbox, so has to be selected manually. To do this, right click the 'Data Flow Sources' group header in the Toolbox window and select the 'Choose items...' option. The system may take a few moments to display the resulting dialog window. From this dialog window select the 'SSIS Data Flow Items' tab and tick the 'mvNET Source' option in the check box list. Click OK and the 'mvNET Source' entry will be displayed in the Data Flow Sources Toolbox group.

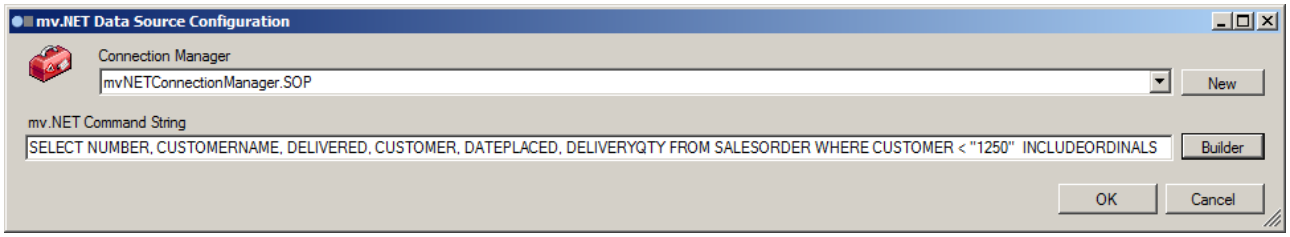
Next we need to add the 'Data Source', so from the 'Data Flow Sources' toolbar drag the 'mvNET Source' to the main designer window.



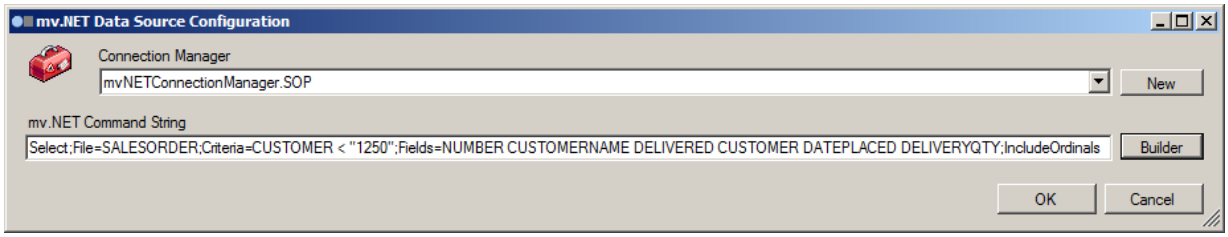
Right click and select the 'Edit' option to display the configuration form:



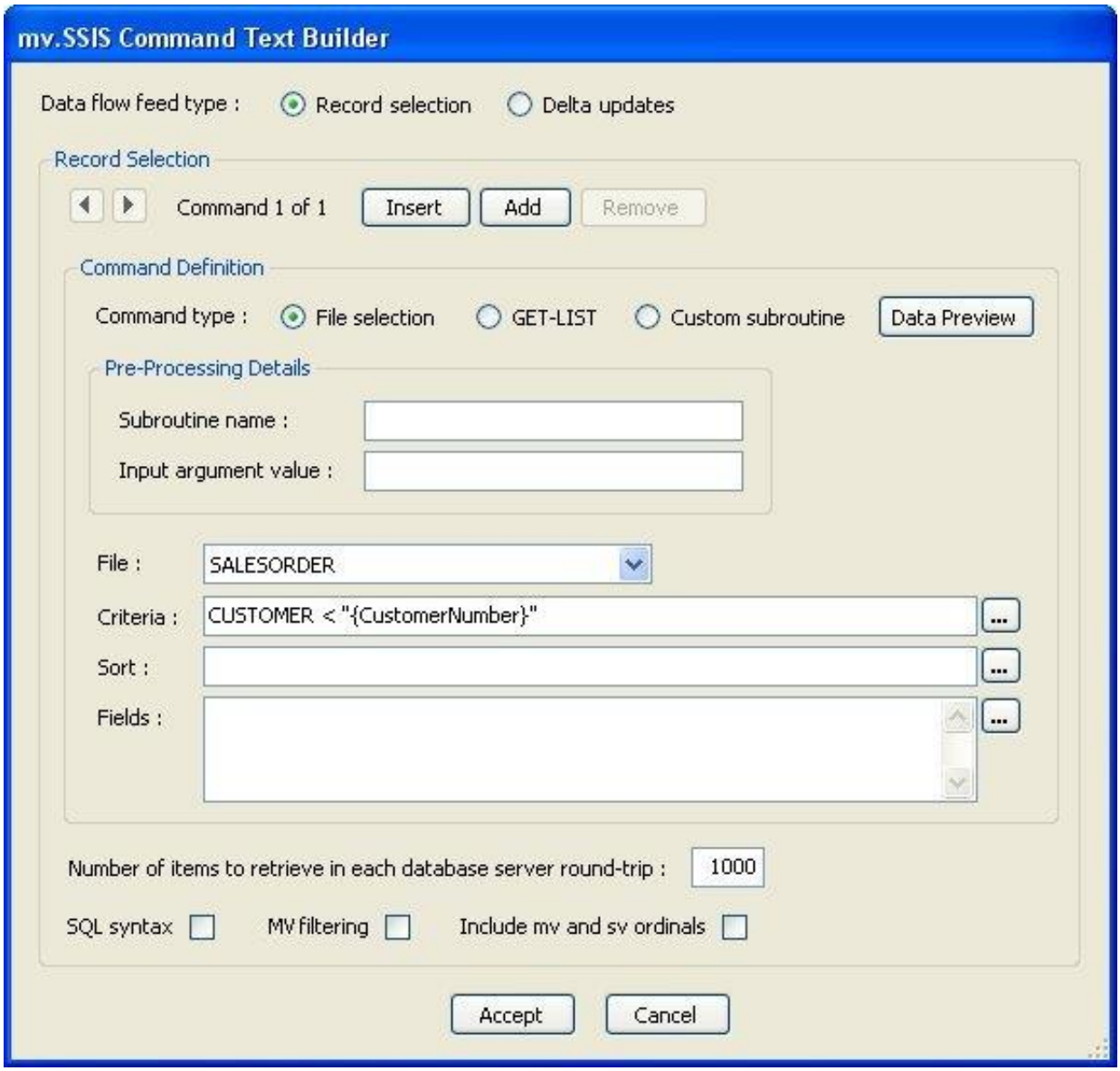
Where you can select the connection manager you added before and enter the command string for selecting the data from the database. This can either be in SQL format or native mv.NET syntax. e.g.:



Or:



Clicking the 'Builder' will take you to the command builder screen which will help you build the command you need:



At the top of the builder form you are able to select the type of feed that this data flow source will produce. For normal data extractions you should select the 'Record selection' option. The 'Delta updates' option is explained in a following chapter.

For 'Record selection' feeds, you are able to define 1 or more commands to be used to extract the required records. For this simple example we will create only a single command.

Each command can be 1 of 3 types, as indicated by the 3 radio buttons at the top of the 'Command Definition' group box. For this simple data transformation we will use the 'File selection' option.

File Selection Command Types

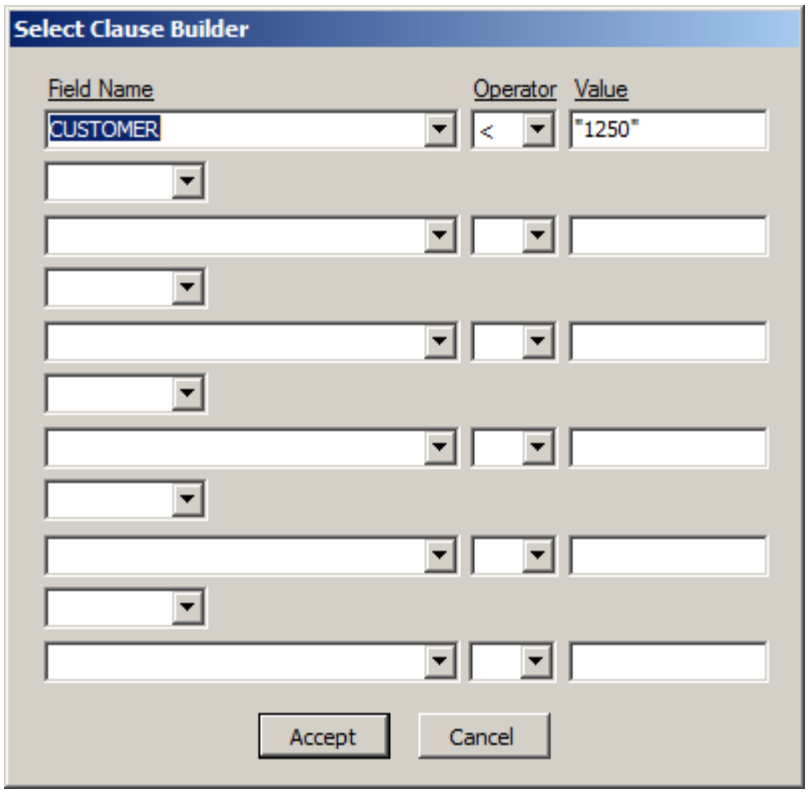
The 'File selection' command type allows you to specify a range of selection details that are to be used to select a list of records and a list of fields from within each of these records from the underlying database.

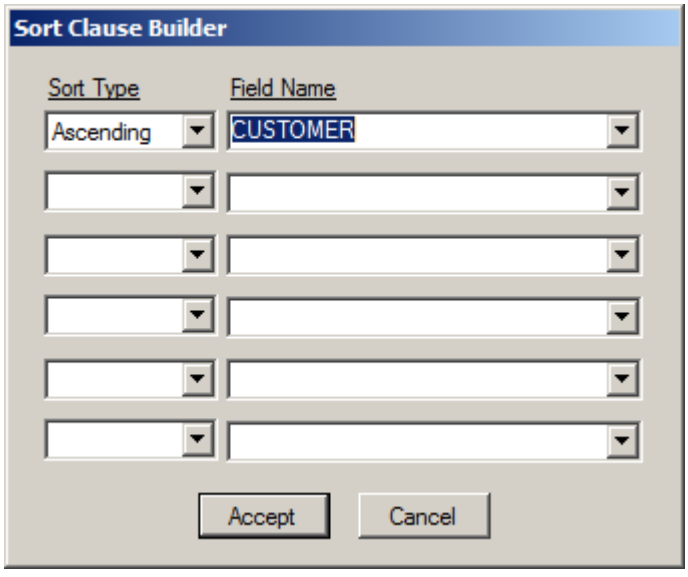
The first thing that can be specified is the name of a database resident subroutine that is to be called before any selection processing is performed. You might use this feature if you need to perform some preparatory work before running the main selection. Any subroutine identified in this input field must accept 3 arguments in its call signature. These arguments are as follows:

Argument #	Direction	Description
1	Input	Call context. This argument will hold the data contained within the 'Input argument value' input field within the command text builder form. The content here may, if required, contain SSIS run-time variables - see chapter on using SSIS run-time variables for the syntax to be used.
2	Output	Item IDs. This argument can be used to return the list of required item IDs. If a non-blank value is returned in this argument it will be assumed to contain an attribute mark separated list of item IDs which will be used as the basis of any subsequent selection processing.
3	Output	Error message. If this argument contains a non-blank value on return from the subroutine it will be assumed that an error has occurred and that this argument holds a description of the error.

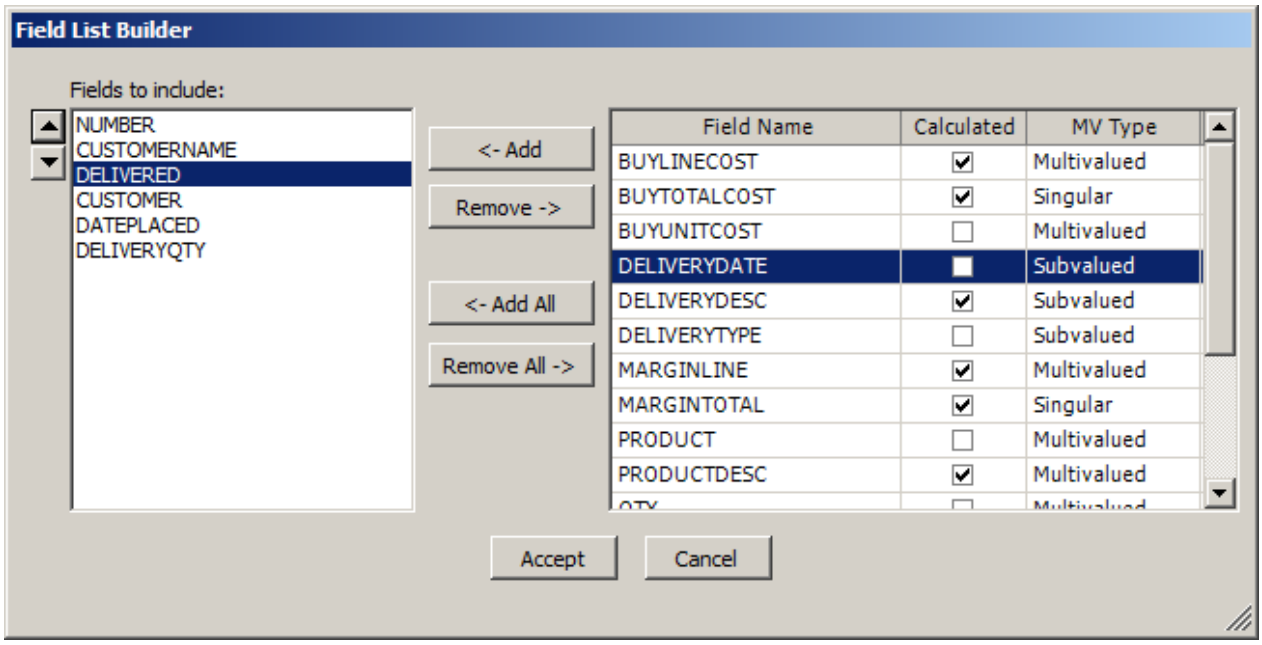
Next, you are able to select the name of the file that is to provide the data.

If required, you are also able to enter selection and sort criteria to control the range and sort sequence of the returned record list. Builder buttons ('...') are provided to assist you in entering these selection and sort clauses:





Finally you are able to specify the fields that are required from each selected record. The field list is list of space separated field names and a builder button is provided to allow you to view and select the available names:



Ordinal Columns

The command text builder window contains a checkbox named 'Include mv and sv ordinals'; this allows you to indicate that you require access to the physical multivalue and subvalue storage positions of nested data fields. This data is presented as separate columns within the SSIS mapping schema. The columns names used here are controlled by the 'Properties' of the relevant file - these can be maintained within the mv.NET Data Manager by right-clicking the file name within the Data Manager's main treeview and selecting the 'Properties' option.

GET-LIST Command Types

The 'GET-LIST' command type is very similar to the 'File selection' type described above except that it uses a SAVE-LIST (as specified by the 'List ID' input field) on the database server to identity the records that are to be selected as opposed to executing a select/sort query. The 'List ID' input field may incorporate an SSIS run-time variable - see chapter on using SSIS run-time variables for the syntax to be used.

When using a GET-LIST command type, you can still call a Pre-Processing subroutine which may be the place where the SAVE-LIST gets created. If a Pre-Processing subroutine is used, the second (Item IDs) argument is ignored.

Custom Subroutine Command Types

The 'Custom subroutine' command type allows you to take complete control over the item selection process by using a bespoke database subroutine. On selecting this command type the display changes to allow you to enter the name of the subroutine and some context

information. The context information can contain whatever string data is required and may also contain SSIS run-time variables - see chapter on using SSIS run-time variables for the syntax to be used.

It is the responsibility of this subroutine to provide both schema and data back to the SSIS framework and can therefore be called in 2 different modes - schema retrieval mode and data retrieval mode. The subroutine must accept 6 arguments in its call signature. These arguments are as follows:

Argument #	Direction	Description
1	Input	Retrieval mode. This argument will hold either the value 'Get schema' or 'Get data' indicating the type of information that needs to be returned.
2	Input	The context (as supplied in the 'Context information' input field). Note, in 'Get schema' mode SSIS run-time variable values will not be present.
3	Input	For 'Get schema' retrieval mode this argument is not used. For 'Get data' mode this contains a value-mark separated list of required field names as returned by the 'Get schema' mode.
4	Input or Output	The file name to be used. In 'Get schema' mode this is an output argument and its value should be set by the subroutine to the file to supply the data. In 'Get data' mode this is an input argument and contains the file name returned by the 'Get schema' call.
5	Output	The return data. In 'Get schema' mode this needs to be set to a value-mark separated list of required field names. In 'Get data' mode this needs to be set to the return record data. The 'Get data' return value must be in the following format: {item} ***sep*** {item} ***sep*** ... That is, a series of item strings concatenated together with a char(30) separating each one. The content of each item string must be the value of each field specified within the content of argument 3 with each field value occupying one attribute position. The entire item string should also have the relevant item ID (or unique string) concatenated at the front as the first attribute irrespective of whether the item ID is specified as one of the required fields or not. For example: `"1":CHAR(254):"Test 1":CHAR(30):"2":CHAR(254):"Test 2"` The above return value returns one field value for item IDs "1"

		and "2".
6	Output	Error message. If this argument contains a non-blank value on return from the subroutine it will be assumed that an error has occurred and that this argument holds a description of the error.

Specifying Fetch-on-Demand Data Retrieval

The retrieval of records from the database server during a transformation is broken down into a series of chunks - each chunk being retrieved by SSIS in a discrete round trip to the database. The 'Number of items to retrieve...' input field on the command text builder form allows you to control how many MultiValue items are retrieved in each such chunk.

Using Mv Filtering

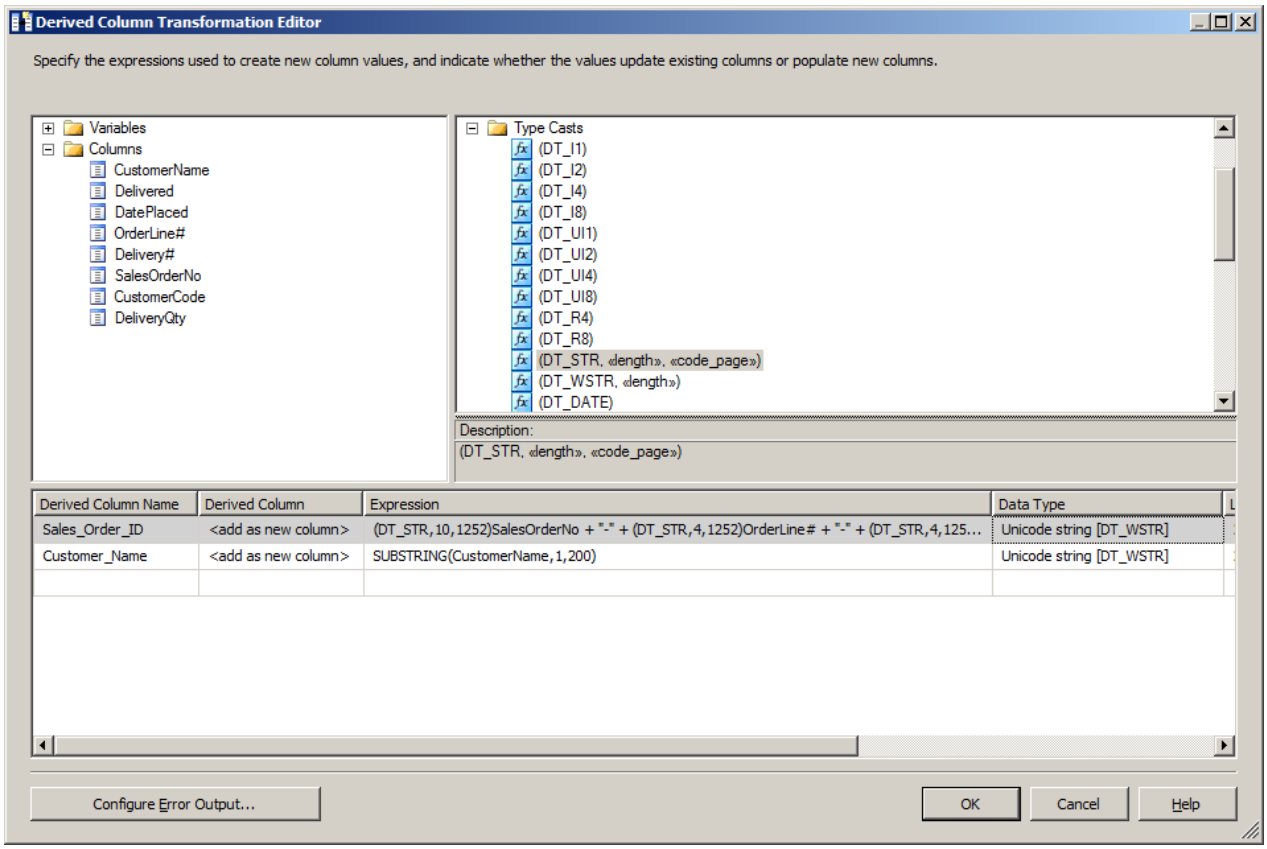
The 'Mv filtering' check box on the command text builder form allows you to specify that selection processing needs to be performed against individual multi/sub values - often referred to as 'exploded selecting' in MultiValue database terminology. This allows a subset of nested data elements within each selected item to be extracted and returned.

Previewing Data

The 'Data Preview' button on the command text builder form allows you to view the data which is going to be passed into SSIS. Once you have entered the details of your commands you may click the 'Data Preview'. If any SSIS run-time variables are referenced within the command text definitions a window will be presented allowing you to enter the values to be used for the variables during the data preview.

Using Derived Columns

Once you have selected the connection manager and defined the selection command text you can then define the target of the data. Normally it is required to perform some sort of transformation especially with string fields which all come across from mv.NET as varchar(4000) fields. So, if we're to target a string to a varchar(20) as the primary key field we could derive the field using the 'Derived Column' transformation as follows:



where our 'Sales_Order_ID' is derived from the 'Sales Order No.' concatenated with the OrderLine sequence and the Delivery sequence in order to generate a unique key :

```
(DT_STR,10,1252)SalesOrderNo + '-' + (DT_STR,4,1252)OrderLine# + '-' + (DT_STR,4,1252)Delivery#
```

Note that the 'OrderLine#' and 'Delivery#' fields are NOT ones we selected...

```
Select;File=SALESORDER;Criteria=CUSTOMER < '1250';Fields=CUSTOMERNAME
DELIVERED CUSTOMER DATEPLACED DELIVERYQTY
```

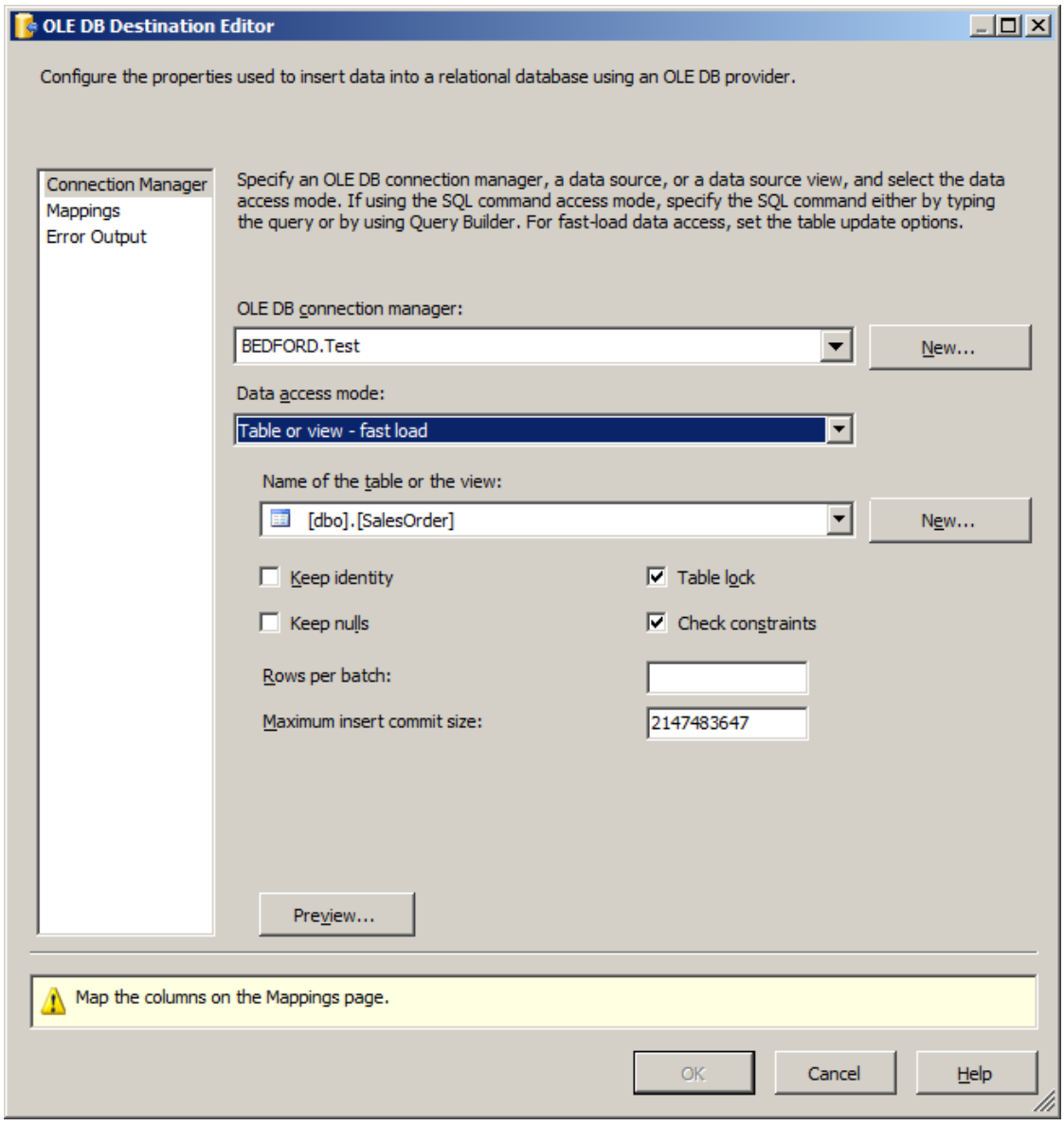
It is part of the result set as a consequence of the system 'flattening' the multi-value dataset into a two dimensional structure, adding the 'OrderLine#' and 'Delivery#' fields to maintain data integrity.

Note: In order to access the 'Derived Column Transformation Editor', you need to click on the Toolbox, drag a 'Derived Column' control onto the Data Flow tab. Connect your 'mvNET Source' control to the 'Derived Column' control so that it appears as input to this item, then right-click on the 'Derived Column' control and select 'Edit'

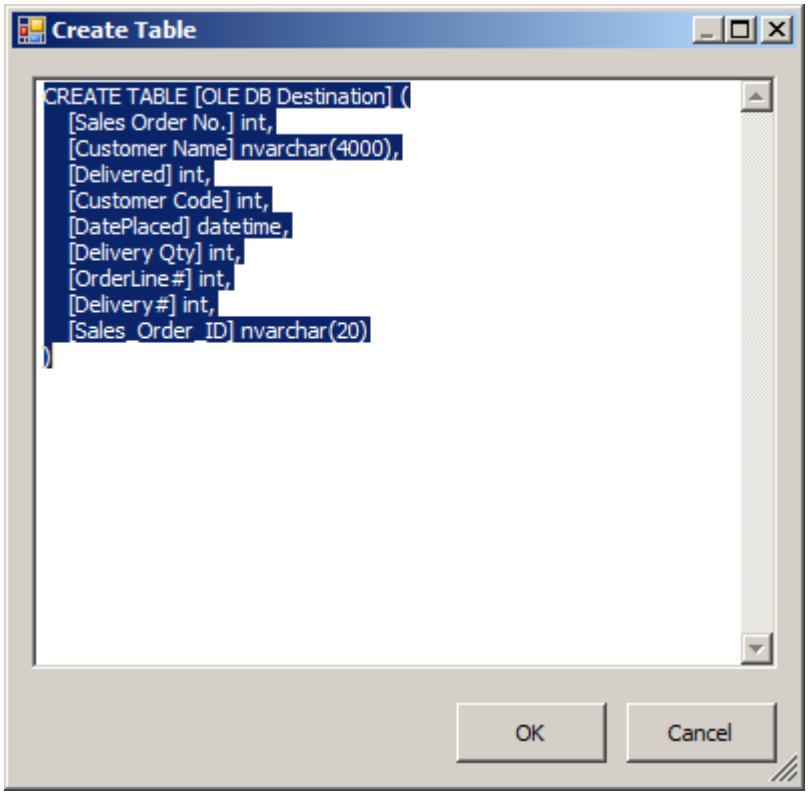
Specifying the Data Destination

We can now add a destination for the data, so drag an 'OLE DB Destination' tool from the 'Data Flow Destinations' toolbox and 'Edit' it using the right-click context menu.

You'll need to add an OLE DB connection manager and connect it to SQL Server, then select the data access mode, in the example below the 'Table or View – fast load' option has been selected and pointed it at to destination table 'SalesOrder'.



It is possible to have the OLE DB Destination Editor generate a default table definition based on the format of the input data by clicking on the 'New' button next to the 'Name of the table or the view' text box:



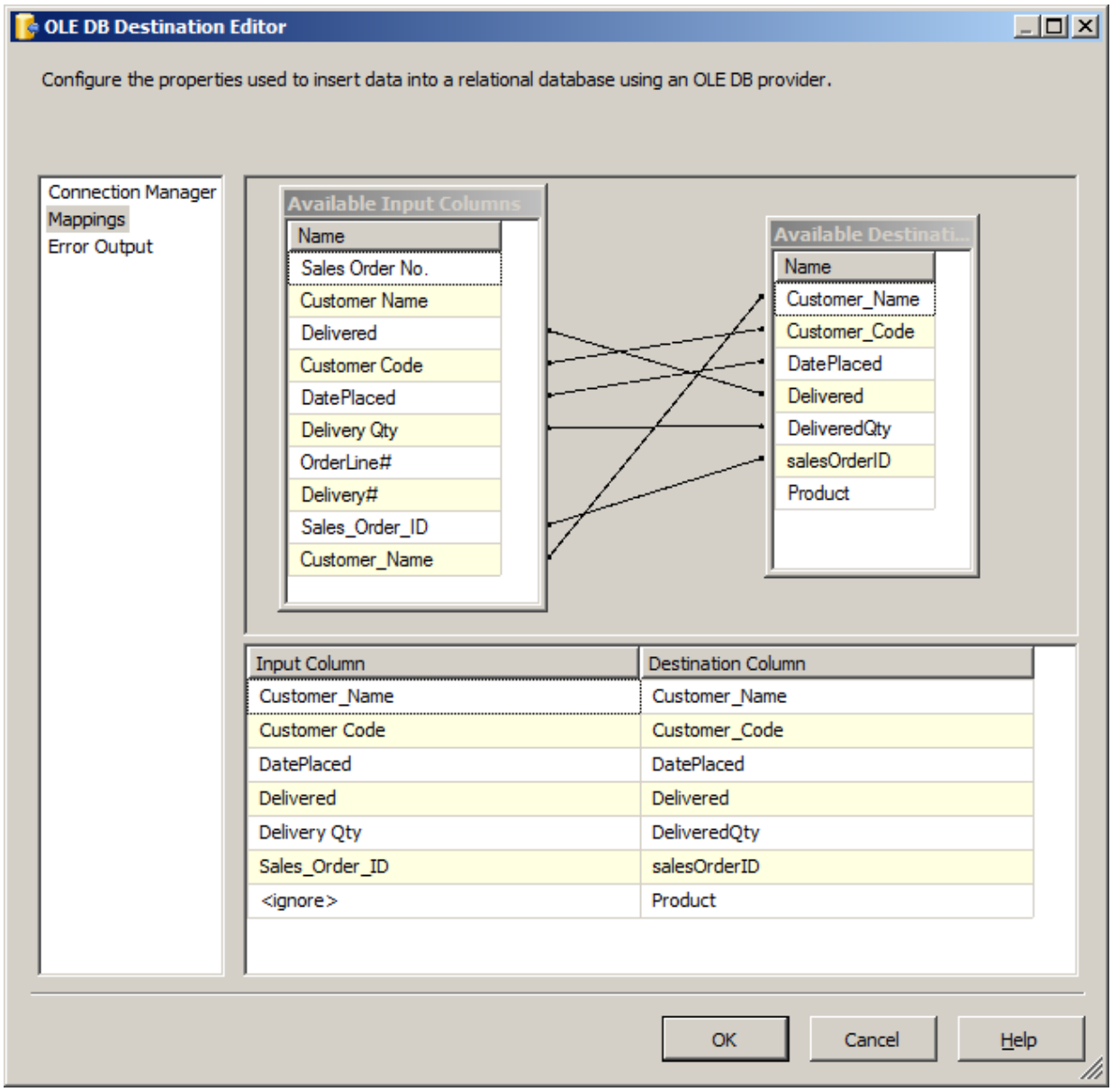
In our case, we have chosen to use the following script to build the 'SalesOrder' table in SQL Server:

```
USE [Test]
GO
/****** Object:  Table [dbo].[SalesOrder]    Script Date: 02/12/2008 10:28:03 ******/
if exists (select * from sysobjects where id = object_id('dbo.SalesOrder') and sysstat & 0xf =
3)
drop table 'dbo'.'SalesOrder'
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[SalesOrder](
[salesOrderID] [nvarchar](20) NOT NULL,
[Customer_Name] [nvarchar](200) NULL,
[Customer_Code] [int] NULL,
[DatePlaced] [datetime] NULL,
[Delivered] [int] NULL,
[DeliveredQty] [int] NULL,
[Product] [nvarchar](400) NULL,
 CONSTRAINT [PK_SalesOrder] PRIMARY KEY CLUSTERED
(
[salesOrderID] ASC
```

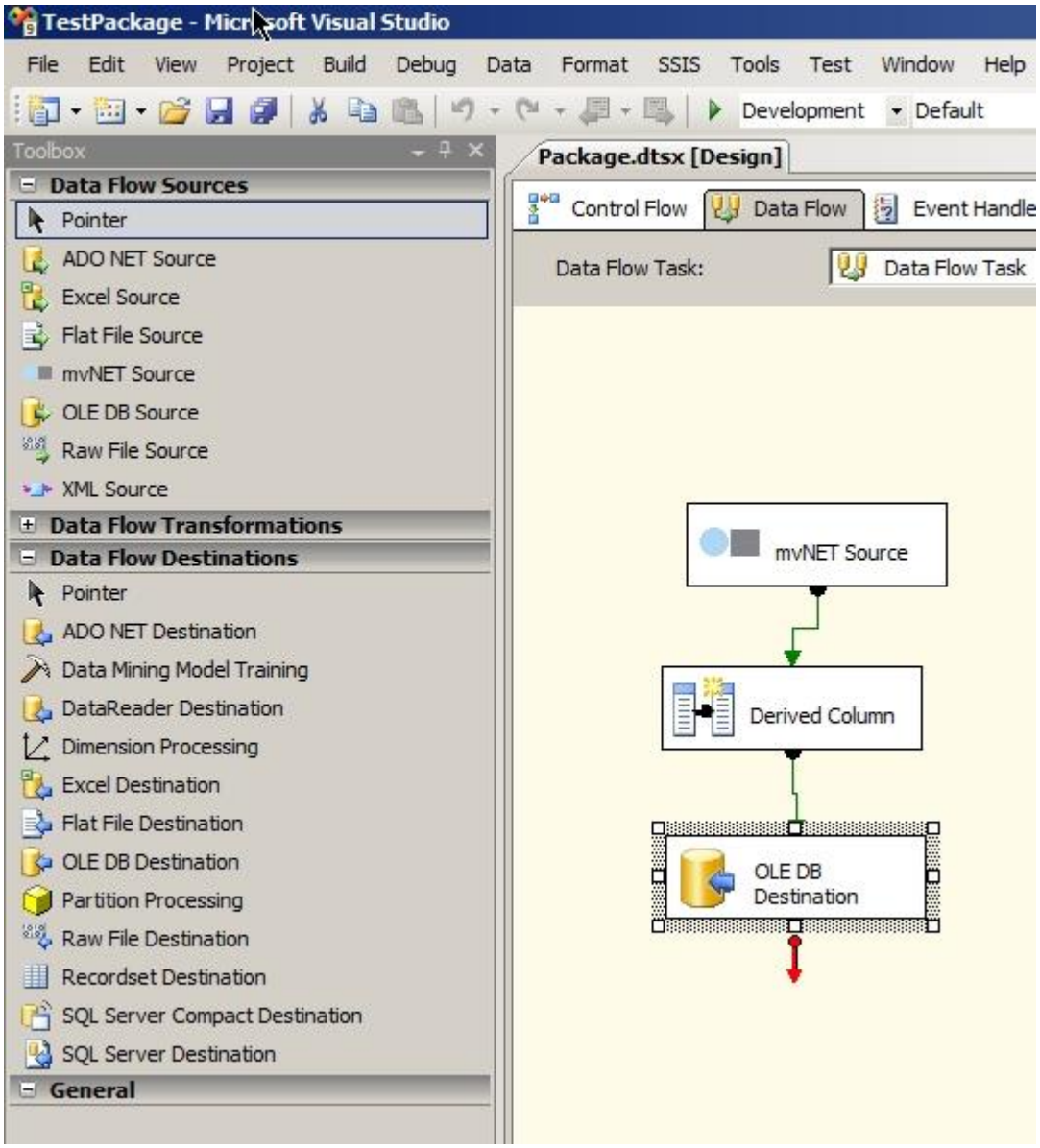
```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS  = ON)
) ON [PRIMARY]
GO
```

Next we need to map the fields from the mvNET data source to the SQL Server table, do this by clicking the 'Mappings' option on the left.

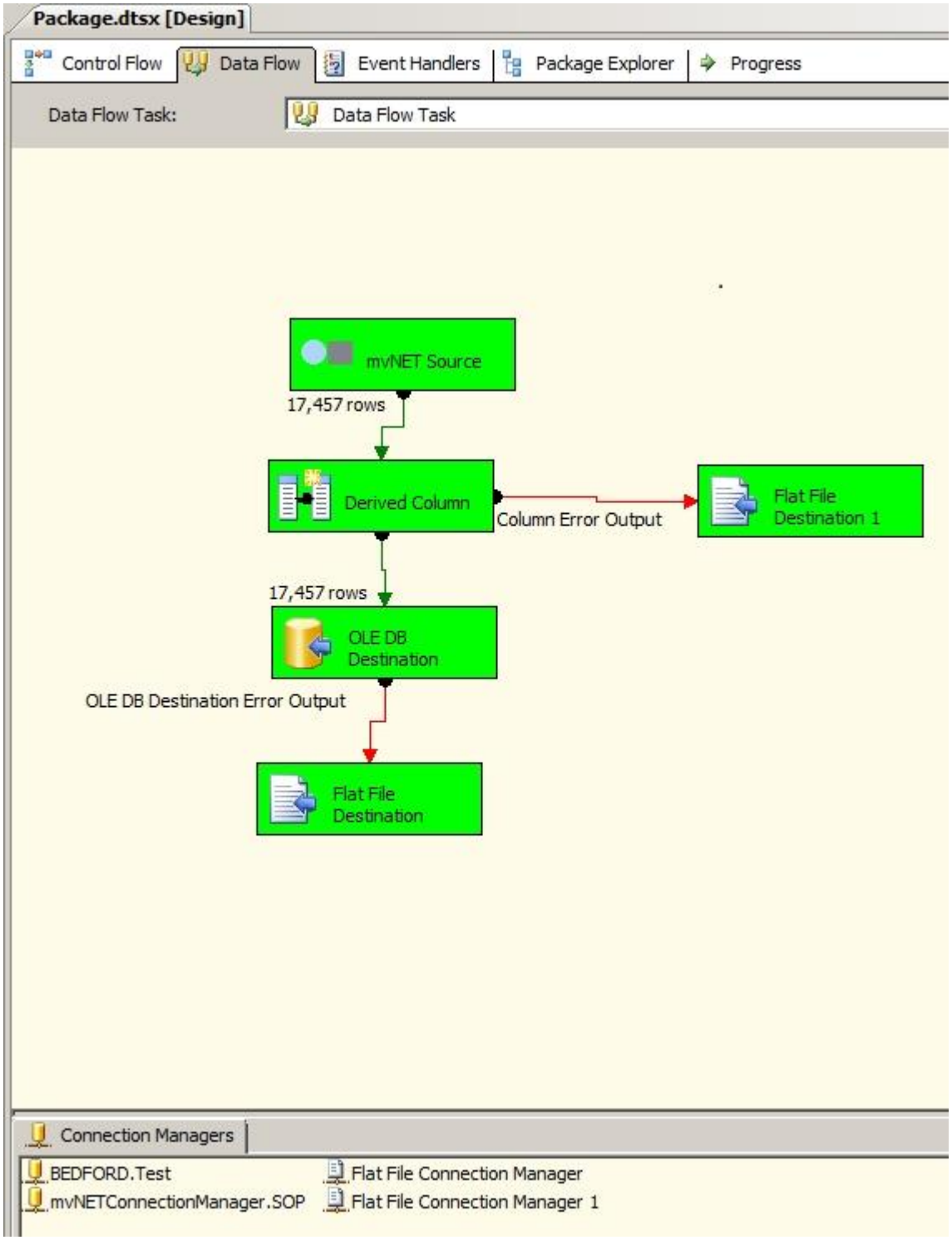
Note that we have mapped the two derived fields 'Customer_Name' and 'Sales_Order_ID' to the destination table rather than the original 'Customer Name' and 'Sales Order No' fields. This prevents warnings appearing about truncating the string data.



Once the fields have been mapped we have a 'Data Flow' process which looks something like this:

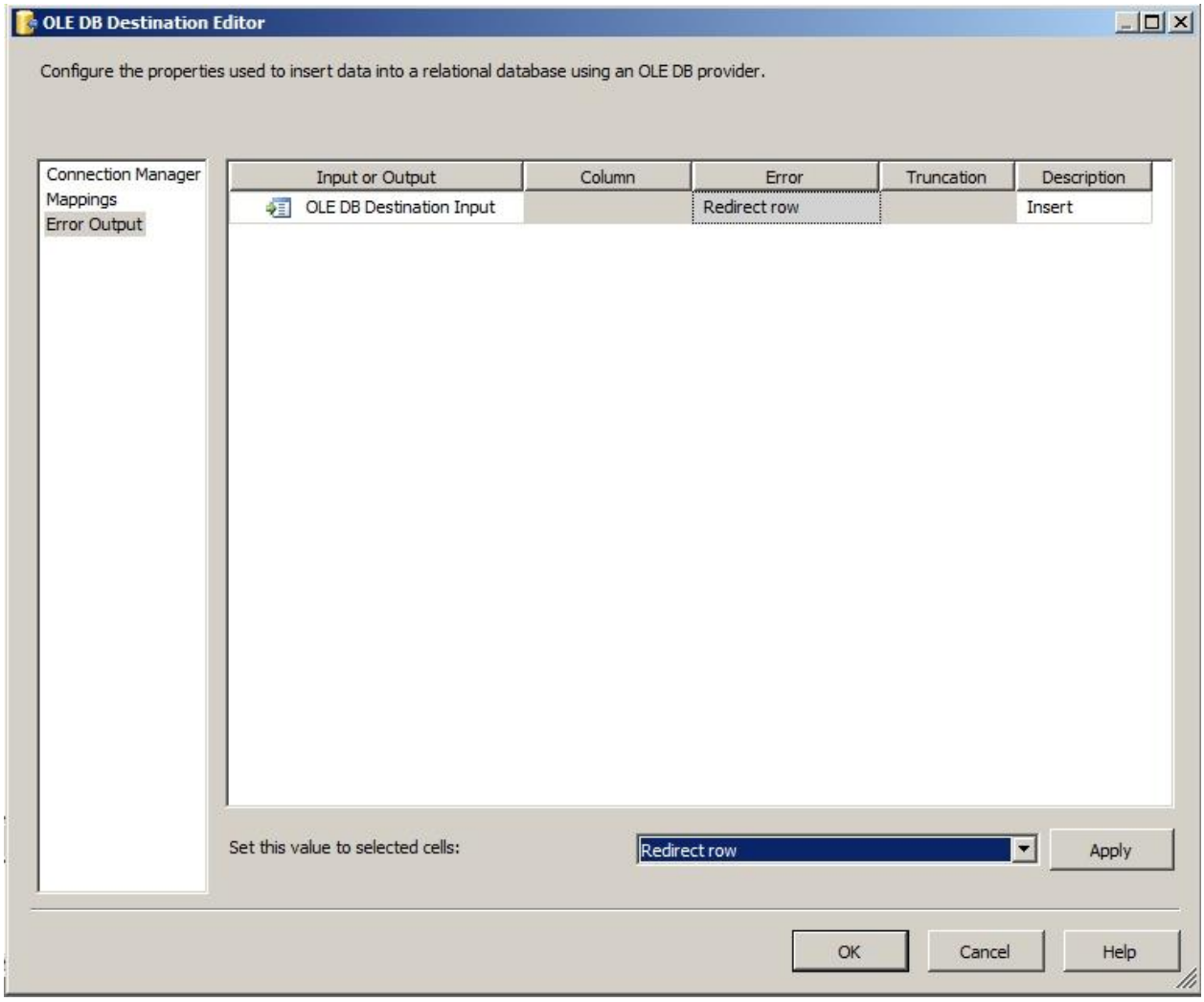


We can now run the package to transfer the data to the SQL Server table:



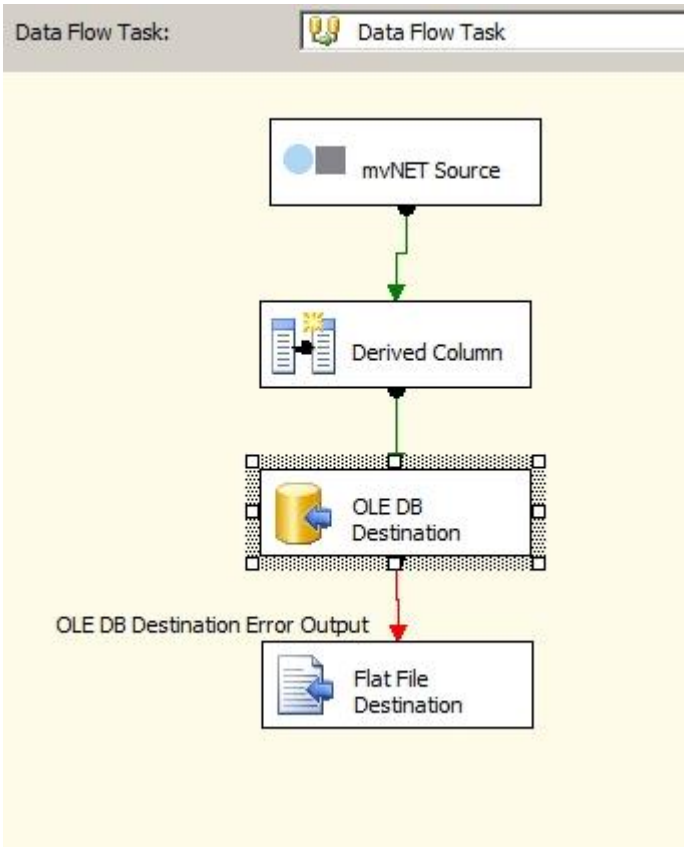
Two Error Outputs have been added in order to capture any errors which might occur either in the generation of the derived columns or in the data to be stored in SQL Server, see below for further details on this.

If errors are encountered during the Data Flow process, especially with the output of the data to SQL Server, it is possible to modify the OLE DB Destination information to include Error Output:

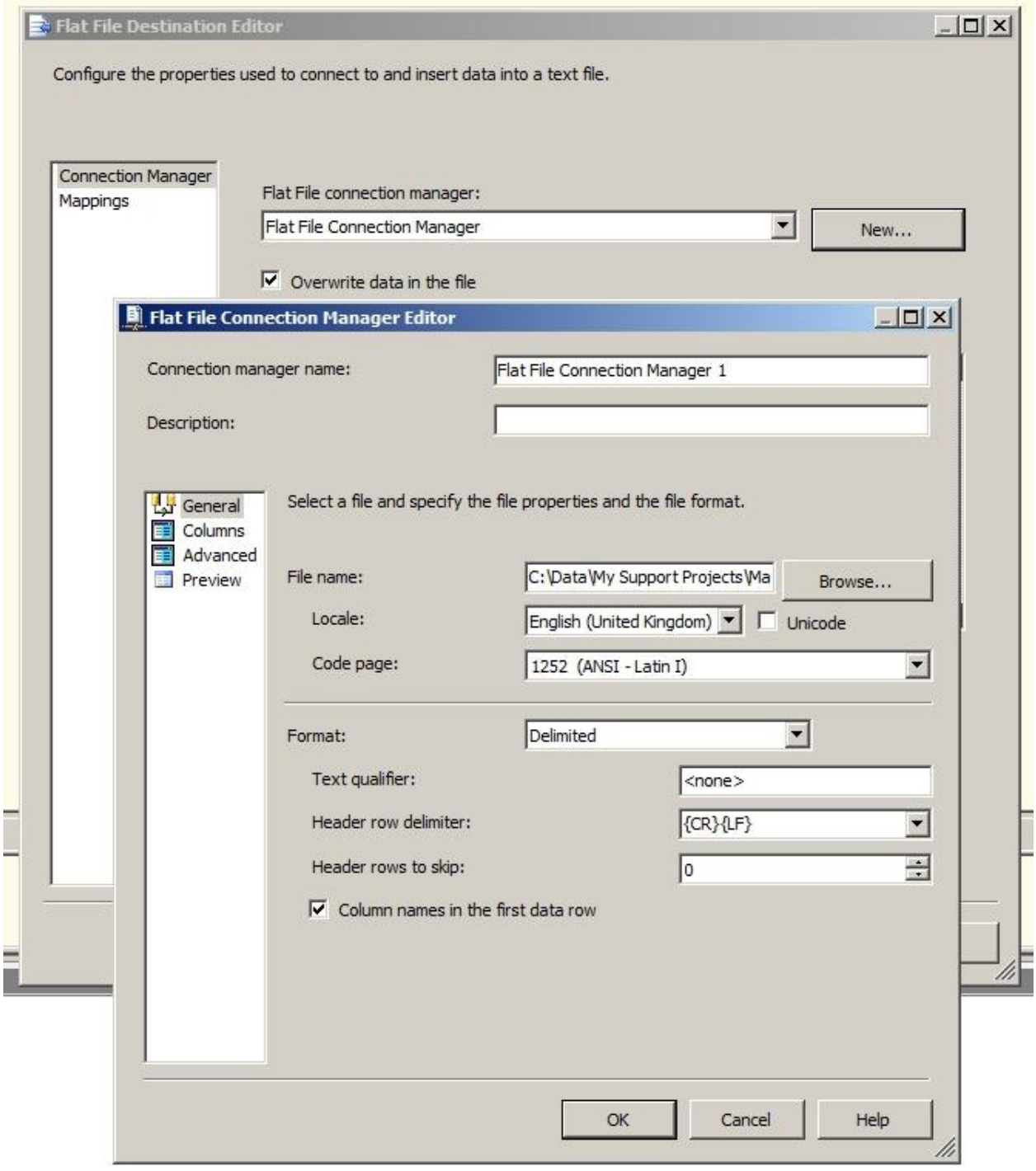


The error output can be redirected to a flat file for further inspection.

A 'Flat File Destination' tool should be dragged from the 'Data Flow Destinations' section of the Toolbox and connected to the 'OLE DB Destination':



We can then proceed to edit the Flat File Destination information in order to indicate where the flat file should reside and what its characteristics should be :



With this in place, all valid data will be written to the SQL Server table, and any data regarded as invalid will be written to the flat file.

Using SSIS Run-time Variables

The SSIS environment allows you to define elements of data that are to be provided at run-time - these are known as run-time "Variables".

This chapter explains how you can use run-time variables within an mv.SSIS data source definition.

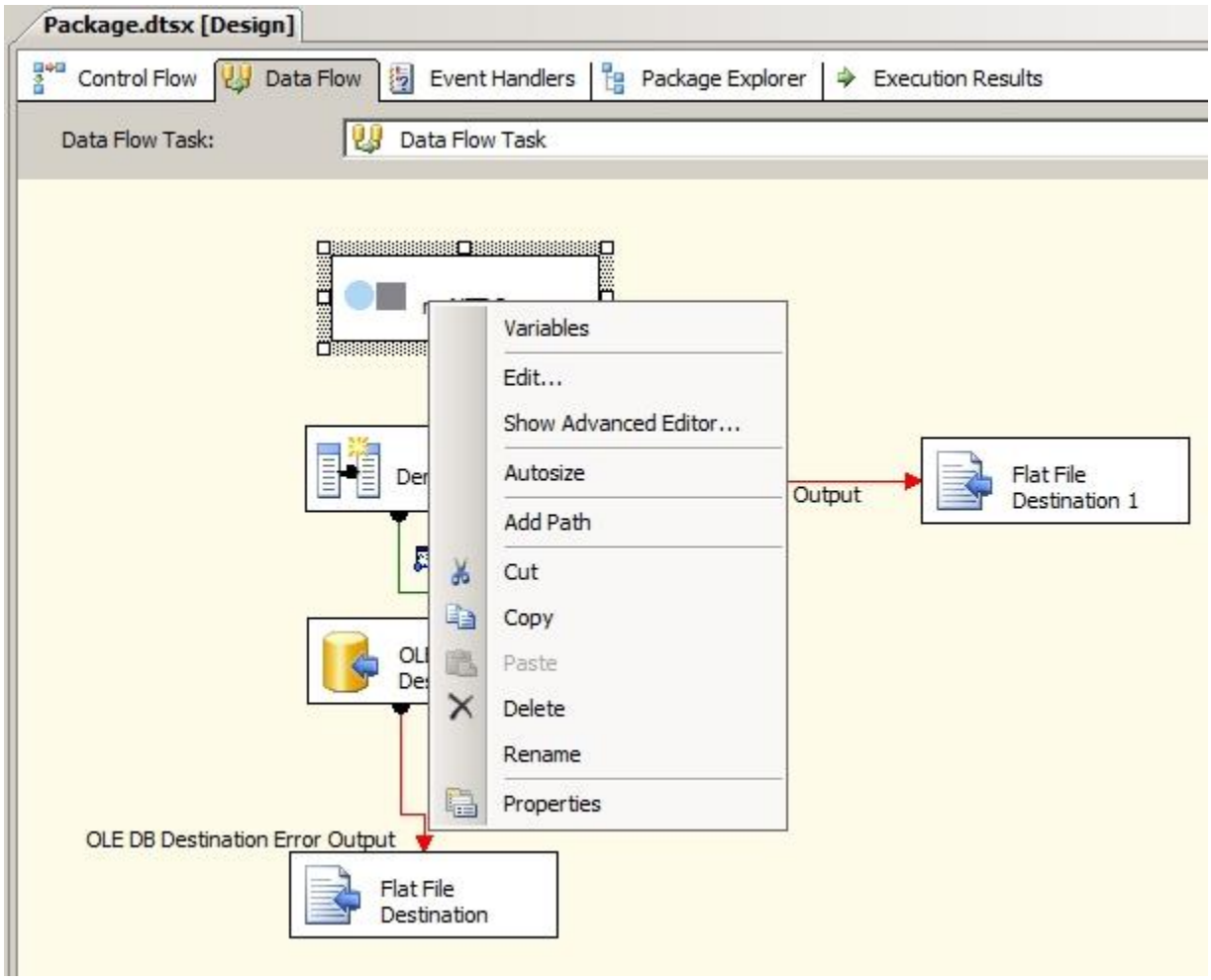
Introduction

If the data that you wish to transfer into SQL Server as part of a transformation package will varying according to one a more pieces of information that are only available at run-time, you will need to make use of SSIS's "Variables" capability. This allows you to define a range of variables that can be referenced within the definition of a package at design time and you are then able to supply the actual values to be used when the transformation is executed.

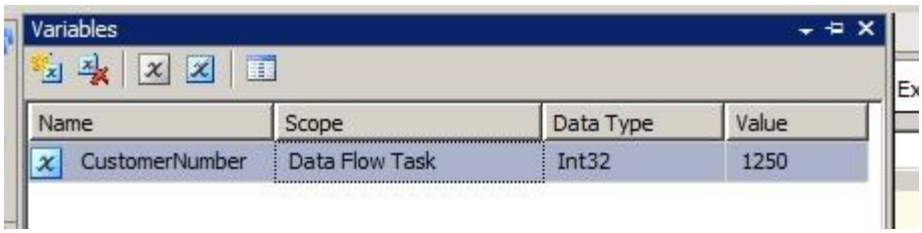
The content of this chapter discusses the topic of "Variables" from an mv.SSIS perspective - i.e. how you may incorporate variables into the mv.SSIS command definition. For further information on defining variables and how to supply their values at run-time please refer to the standard SSIS documentation.

Defining Variables

Variables can be defined by either right-clicking on a particular Data Flow Component or on the design surface of the package and selecting the “Variables” option from the drop-down menu



The “Variables” designer allows the variables to be created, modified and deleted when in Design Mode:



SSIS allows you to create a range of different variable types as well as provided a selection of standard pre-defined variables which are automatically available within any package.

Referencing Variables

Variables may be referenced in a number of places within an mv.SSIS data source definition:

- GET-LIST ID
- Selection clause
- Sort clause
- Subroutine call context

In order to reference a variable in any of these places you need to place the relevant variable name within curly braces - note the casing of the variable name **is** significant. For example, if you have created a variable called "CustomerNumber", you could reference it as part of your selection clause as follows:

```
WITH NUMBER = "{CustomerNumber}"
```

Using Delta Updating

mv.SSIS allows you to restrict the data that is transferred as part of a transformation package to only that information which has been added, amended or deleted. This type of data restriction is termed 'delta updates' – or 'net changes' in some circles.

This chapter explains how you can use the delta updating feature of mv.SSIS.

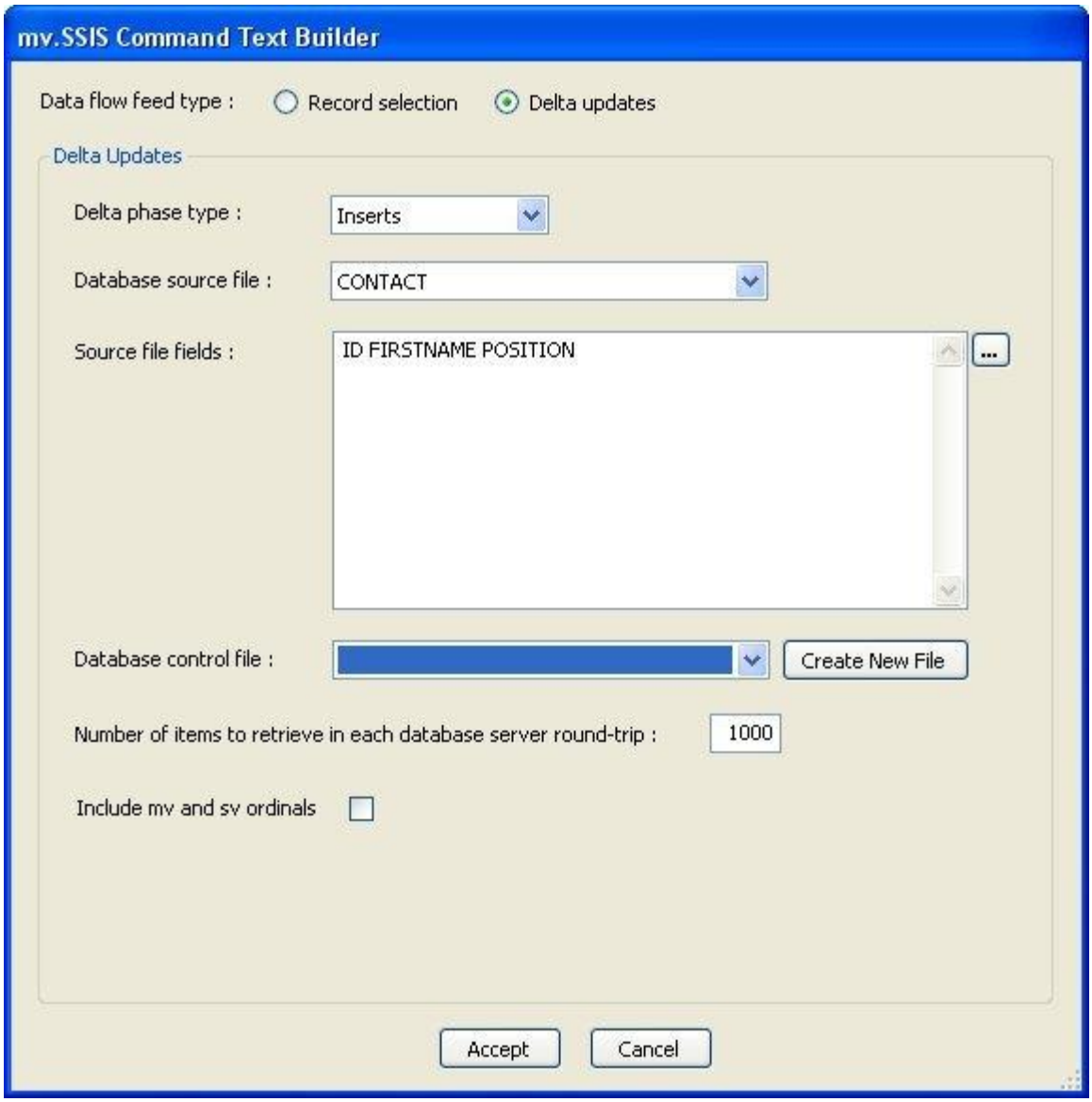
Introduction

If you have very large database files residing on your source MultiValue database, it may be impractical to transfer the entire content of such files on a regular basis. Therefore, the obvious alternative to this repeated mass transfer is to only bring across the data that has changed.

mv.SSIS has a built in feature called 'delta updates' which is designed for this very purpose and provides a flexible infrastructure for you to control which types of data amendments (inserts, updates and deletes) are transferred as part of a package.

Delta Updates Basics

As illustrated in the previous chapter, one of the 'Data flow feed type' options available at the top of the mv.SSIS Command Text Builder window is 'Delta updates' – you need to select this option in order to use delta updating. Selecting this option changes the appearance of the command text builder window to the following display:



The first 'Delta phase type' input field allows you to identity the type of delta updating phase (or "feed") that will be performed/supplied by this particular data flow task component. The following options are available:

Prepare: This should be the first delta phase performed in the transformation. It performs the initial data comparison task in order to identity which data records on the MultiValue database have changed. This phase does not return any data, it merely creates control data on the MultiValue database in readiness for the execution of further delta phases.

Updates (Delete): This phase extracts the primary keys of those records that have been updated. This particular feed allows you to remove the data on the SQL database that relates to the updated records in readiness for the insertion of the new version of the record data.

Updates (Insert): This phase extracts the data of those records that have been updated. This particular feed allows you to insert the new version of modified record data into the appropriate SQL database table(s).

Deletes: This phase extracts the primary keys of those records that have been deleted. This particular feed allows you to remove the data on the SQL database that relates to the deleted record.

Inserts: This phase extracts the data of those records that have been inserted (added) to the MultiValue database file. This particular feed allows you to insert the new data into the appropriate SQL database table(s).

From the above descriptions of the various phases available, it can be seen that the basic principle involved with delta updating is that you prepare for the data transfer by performing the preparatory data change analysis phase. You can then pull over the data change types (inserts, updates and deletions) according to the nature of the synchronization that is in force between the MultiValue database and the SQL server. You do not need to perform all of the phases but you do need to perform the initial Prepare phase first.

If you need to update multiple SQL tables with the same data feed, a particular phase may be executed multiple times in order to make the same data feed available for multiple uses.

Delta Update Specification Details

The Command Text Builder window allows you to enter the details of a particular delta updating phase. The following input fields are presented:

Delta phase type: Allows you the select the delta updating phase type.

Database source file: Allows you to select the MultiValue database file which is to supply the data.

Source file fields: Allows you to specify which fields are to be retrieved from the source file. This input field is disabled for some of the delta phase types.

Database control file: Allows you to select the MultiValue database file which holds/is to hold the data change analysis information. This file must be structured in a particular way and must therefore always be created initially using the 'Create New File' button to the right of this input field – see below

Number of items to retrieve ...: For large data transfers, the overall transfer task will be 'chunked' into a series of database server roundtrips. This input field allows you to specify how many database items are to be retrieved from the database server in each of these roundtrips.

When the 'Create New File' button is clicked, the following window is displayed:

This window allows you to specify the name of the new control file along with the approximate number of items that are contained within the source file specified in the Command Text Builder main window. The 'Control history size' allows you to specify the number of previous transfers that are to have their details retained within the control file.

The mv.SSIS framework allows you to run the data change analysis based on any previous analysis run (see section below for more details on this) therefore, this option allows you to retain a certain number of previous analysis details on-line ready for use. When the history size is exceeded, the control information for the oldest run is deleted from the MultiValue database.

Include mv and sv ordinals: Allows you to indicate that you require access to the physical multivalue and subvalue storage positions of nested data fields. This data is presented as separate columns within the SSIS mapping schema. The columns names used here are controlled by the 'Properties' of the relevant file - these can be maintained within the mv.NET Data Manager by right-clicking the file name within the Data Manager's main treeview and selecting the 'Properties' option.

The Delta Update Mechanism

It is perhaps useful for those people using the delta update feature to understand how it performs its task. This understanding is also useful if you want to take control of the initial preparation phase (see below).

The 'Prepare' delta update phase is a 2-pass process. The first pass traverses the source file content and compares this data content with the snapshot taken in the previous preparation phase. This allows the preparation phase to identify those items that have been either inserted or updated. The second pass then traverses the content

of the snapshot of the previous preparation phase and from this is able to determine which items have been deleted.

The preparation phase writes data to 3 reserved data portions in the control file: INSERTS, UPDATES and DELETES. All items written to these files are empty (only the item ID is used) and each data portion indicates the items that have been affected (in the way indicated by the name of the data portion) since the execution of the previous preparation phase. Execution of subsequent delta phase types uses the information in these 3 data portions.

If you wish to take control of the preparation phase, (i.e. information indicating which items have been modified exists somewhere in the database and can be used as a quicker alternative to traversing the content of the source file) you can create the content of the INSERTS, UPDATES and DELETES data portions manually. The 'Custom subroutine' record selection type data flow task could be used for this purpose.

Using the Delta Updating Data Feeds

Each of the delta update phase types (with the exception of the Prepare phase) supplies a data row source that can be passed into an SSIS component. The data contained in the feed will be either (for deleted data) a single column containing the item ID of the deleted record, or (for inserted or updated data) will contain all of the data fields specified in the delta update phase definition. If you specify fields that contain nested data within the fields list of the delta phase definition, the data feed will supply a series of ('exploded') first normal form rows representing the full record content.

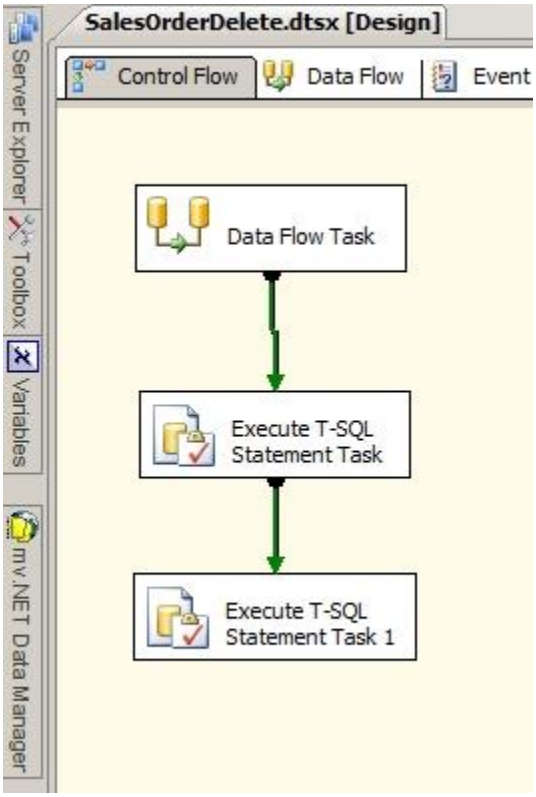
Processing Deleted Data

When an item is deleted from the MultiValue database, its ID will be contained within the 'Delete' delta phase data feed. This (obviously) indicates that the row (or rows in the case of nested data) in one or more SQL tables will need deleting.

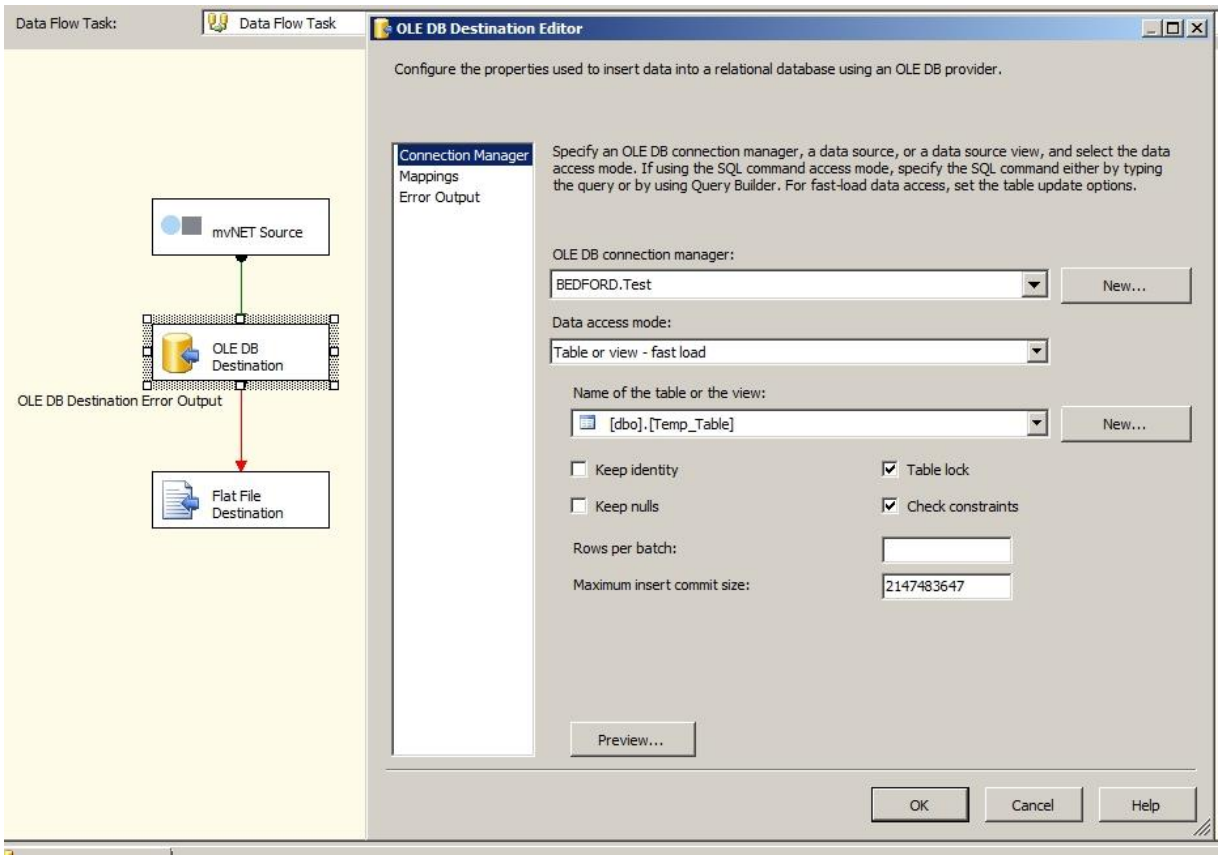
A convenient, efficient and easy to implement way of handling this scenario is to create a table within the SQL database that contains an INSERT trigger. The output from the delta update Delete phase can then be directed to this table and the trigger can then execute a delete statement against the required table incorporating a Where clause using the supplied item ID. Using this technique you can create more than one column within the trigger table, one for each table that requires handling in this manner within the SQL database.

An alternative to the above trigger-based technique is to write the primary keys to a temporary table and then use the keys in this table to select and delete items from the main database table. Once these primary keys have been used, they too can be deleted.

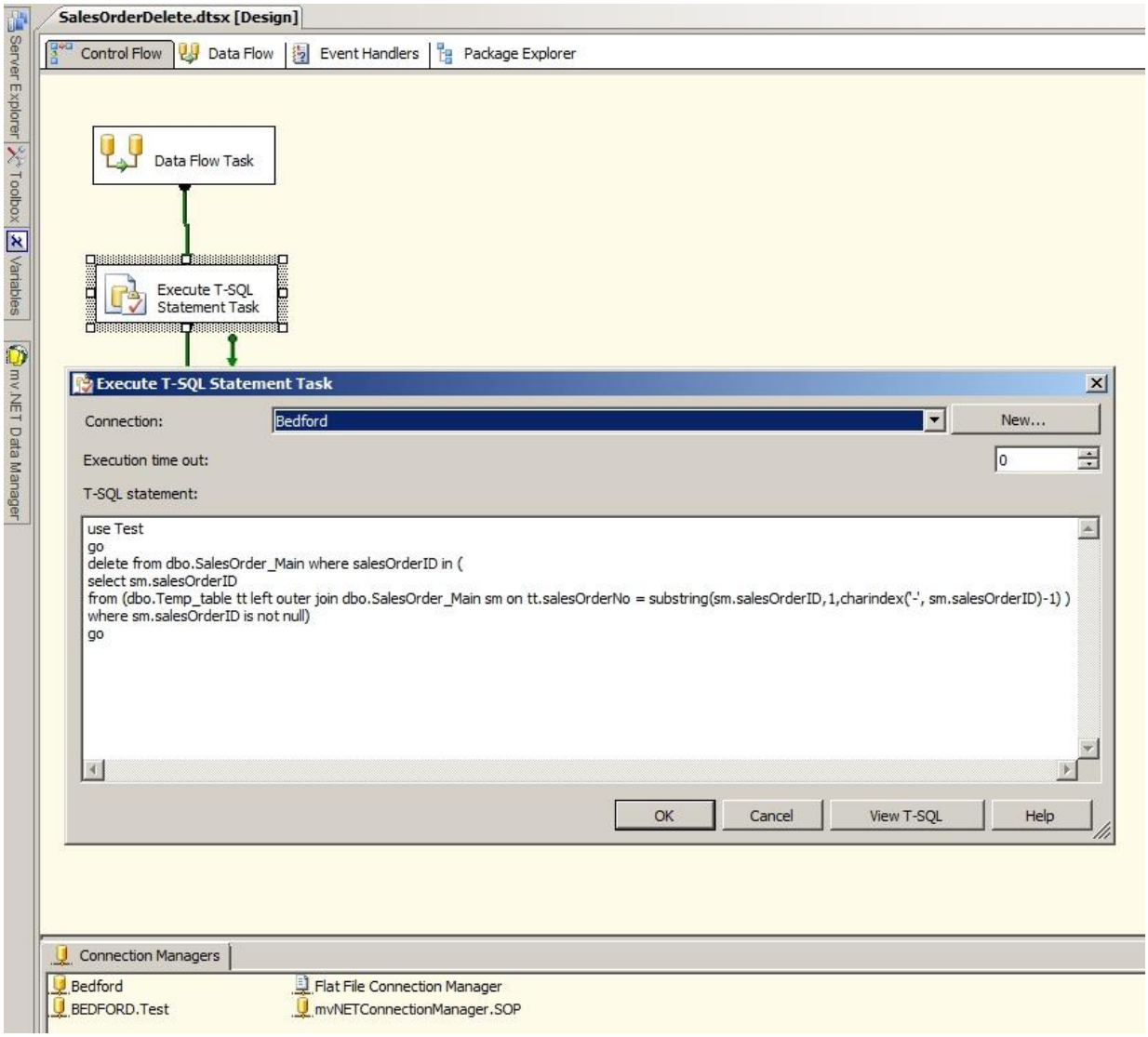
This technique requires a number of discrete Data Flow tasks to achieve the end result:



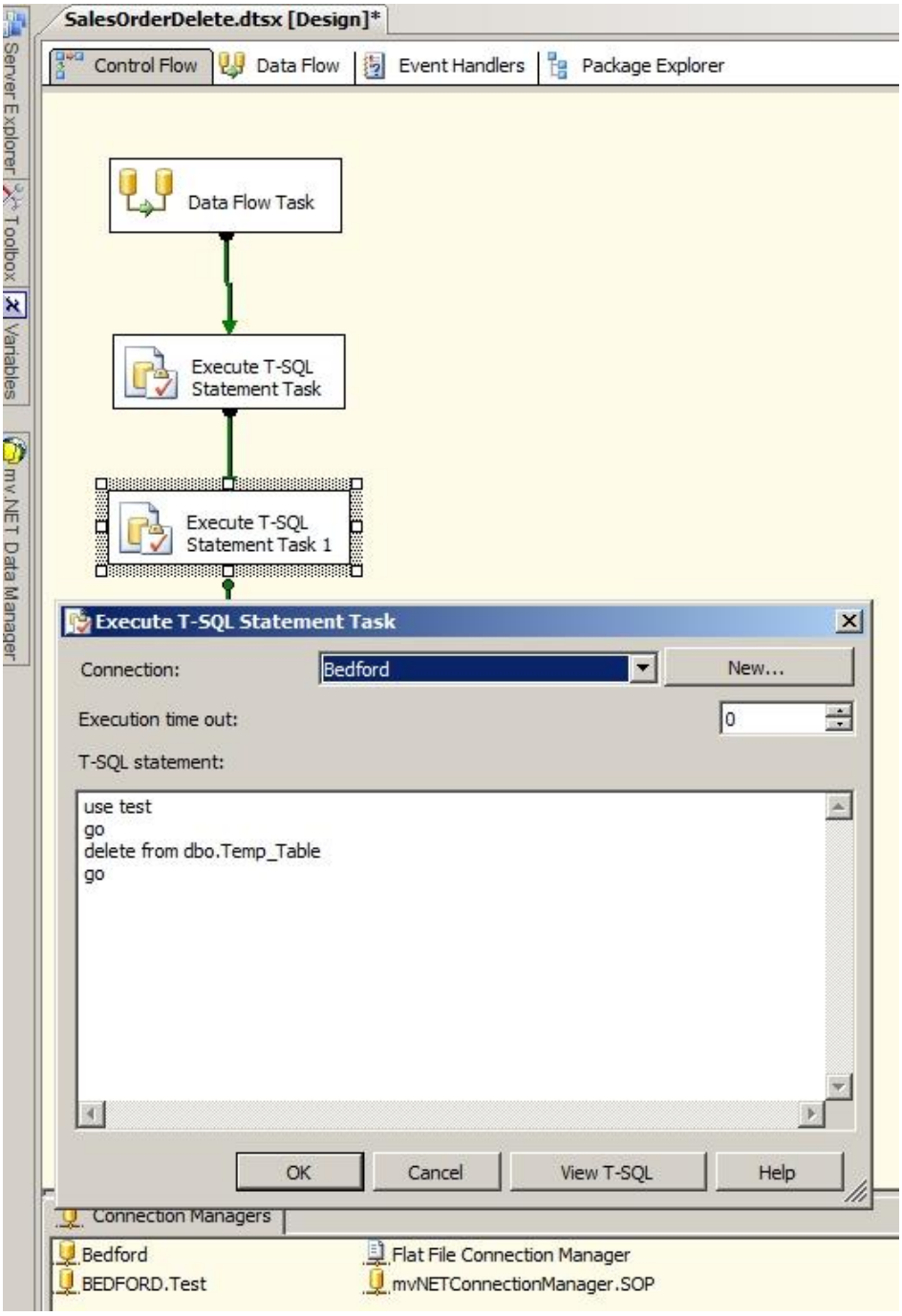
The first Data Flow Task writes the primary keys to a temporary table:



Using the second Data Flow task, the primary keys in the temporary table can then be used to delete items from the main database table. The keys in the temporary table are in the format “678”, and the keys in the main database table are in the format “678-1-1”, “678-1-2” etc.



On successful deletion from the main database table, using the 3rd Data Flow task the keys in the temporary table can also be deleted:



Processing Updated Data

Processing updated records is a 2-part process, hence there are 2 separate feeds available - "Updates (Delete)" and "Updates (Insert)".

The first feed allows you to remove the old copy of the data which has been updated from the SQL database and can be used in exactly the same way as the standard "Deletes" data feed. Note, even If the data item that has been deleted from the MultiValue database contains nested data, only a single entry will be provided by this feed. You therefore need to remove all data that relates to this primary key.

The second feed allows you to insert the new data and is presented as one or more rows, each containing a series of columns (as defined in the mv.SSIS data source "Fields" input field). You may use this second feed as a standard SSIS row feed, directing the data into the relevant SQL table. If the data item that has been amended in the MultiValue database contains nested data, a series of normalized rows will be presented by this feed.

Processing Inserted Data

When an new item is added to the MultiValue database it will be presented as one or more rows, each containing a series of columns (as defined in the mv.SSIS data source "Fields" input field). If the data item that has been inserted into the MultiValue database contains nested data, a series of normalized rows will be presented by this feed.

The Delta Update Summarized

From the above it can be seen that delta updating really boils down to 3 main things:

- The Preparation phase. That is, analyzing what data has changed
- Processing rows to be deleted
- Processing rows to be inserted

This makes the delta updating feature of mv.SSIS a relatively simple yet very flexible architecture which is able to cater for a wide range of updating scenarios.

Product Licensing

This chapter provides a description of the concepts of mv.SSIS licensing and explains how you can request and register product licenses.

Summary of mv.SSIS Licensing

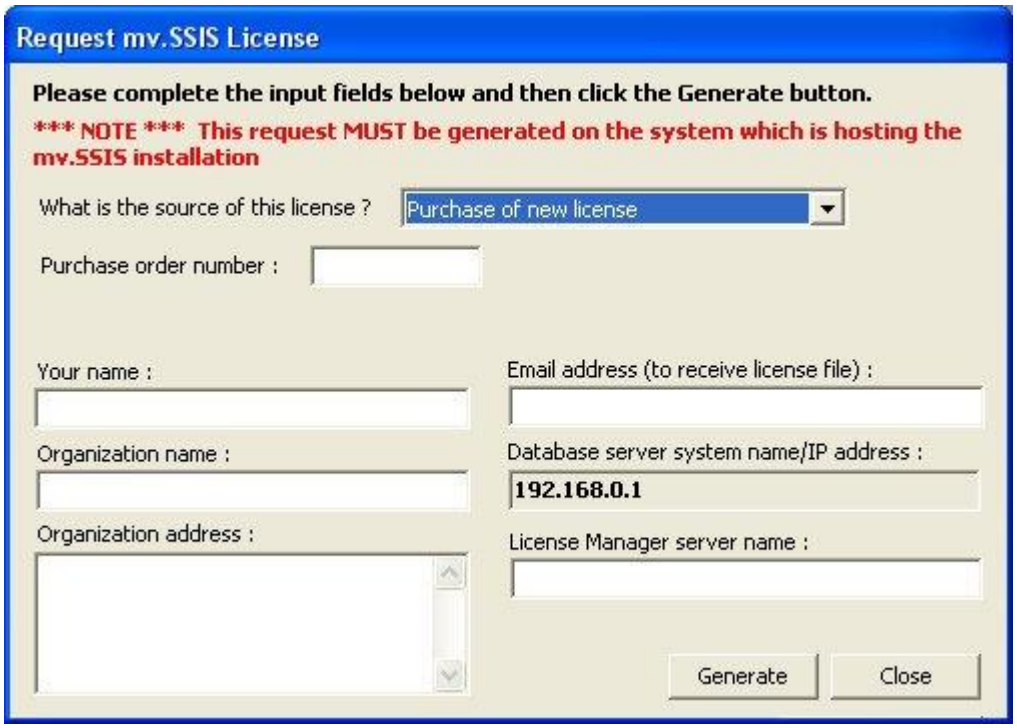
mv.SSIS is licensed per MultiValue database instance and each mv.SSIS license is keyed into a physical system running SSIS.

Requesting mv.SSIS Licenses

In order to receive an mv.SSIS license you first need to request it. This can be done using either the Data Manager (mvNET.DataManager.exe) .

NOTE, when generating requests for mv.SSIS licenses, the Data Manager MUST be run on the system which is going to or which is currently running the SSIS installation.

Using the Data Manager, you need to right-click the appropriate server profile and select the 'Request mv.SSIS License' option. On doing this, the following window will be displayed:



The mv.SSIS License Request Window

On entering the appropriate details into the above input fields, you should click the 'Generate' button in order to be shown the text which you should email to your mv.SSIS supplier in order to receive the relevant license.

Installing Licenses

Once a product license has been requested, you will receive an email containing a license file. On Vista or Windows 2008 server systems, this file should be saved into the following location:

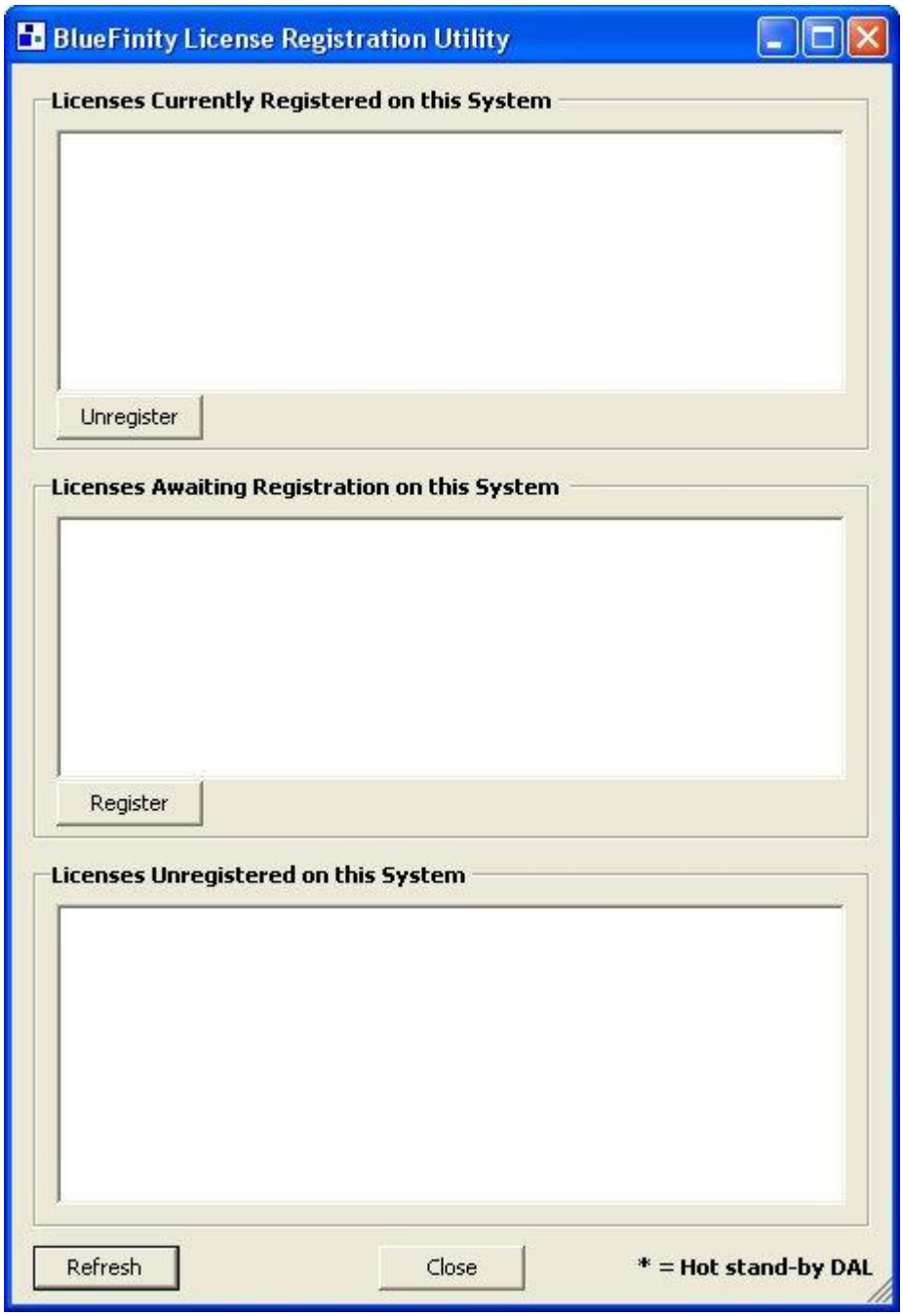
C:\ProgramData\BlueFinity\Licences

On all other systems, this file should be saved into the following location:

C:\Documents and Settings\All Users\Application Data\BlueFinity\Licences

Note, you need to perform this on the system which is hosting the SSIS installation.

After saving the license file, please run the License Registration Utility – a shortcut to which will have been placed on your Start\BlueFinity\mv.NET menu by the mv.NET setup routine. This registration utility displays the following screen:



The License Registration Utility Window

If you have already have a previously registered license for the same database (for example a previously activated evaluation license), you will need to highlight it within the top listbox and click the Unregister button to remove it. Next, highlight

the new license within the Licenses Awaiting Registration listbox and click the register button. At this point your new license will be active.

NOTE, it is essential that you do not amend the contents of any license files.

mv.SSIS Evaluation Mode

When the mv.SSIS product is first installed it will, by default, automatically install a 30 day evaluation license. Any purchased permanent licenses applied during this evaluation period will, naturally, override this behavior. After the 30 days evaluation period has expired, usage will only be allowed if the appropriate permanent license is registered.