A Stock Standard ETL process using SSIS

By Allan Ford

Date: 12/12/2018

This document is mastered as per:

f:\ajf\dongle\FORAL USB\web\A\_Stock\_Standard\_ETL\_Process\_Using\_SSIS.htm

This document is on the web as per:

<http://members.iinet.net.au/~allanford17/A_Stock_Standard_ETL_Process_Using_SSIS.htm>

Discussion

(with some thanks to Figure 11-1 of book “SQL Server Integration Services Design Patterns”)

I wanted to cater for:

-          Inserts to source table to be reflected in the destination table,

-          Updates to source table to reflected in the destination table, but only process destination table row updates where one or more source attribute values differs. (i.e. avoid “blind updates”),

-          Deleted records from source table to be reflected in the destination table,

-          Use a data compare type of method. (i.e. not using triggers and table event logging on source table)

-          Cater for attributes (dates, string, number) and cater correctly for null values.

-          Cater for error handling .. e.g. the insert to destination may cause an error due to a table constraint e.g. a foreign key constraint or column value range constraint.  Likewise the update may cause an error case or the delete may fail due to child records or some reason

Database Table definitions (for this example)

USE [foraldb]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Table\_1]    Script Date: 12/12/2018 12:26:11 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Table\_1](

       [ID\_COL] [int] NOT NULL,

       [NAME\_VAL1] [varchar](50) NOT NULL,

       [DATE\_VAL1] [date] NULL,

       [NUMBER\_VAL1] [float] NULL,

 CONSTRAINT [PK\_Table\_1] PRIMARY KEY CLUSTERED

(

       [ID\_COL] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

CREATE TABLE [dbo].[Table\_2](

       [ID\_COL] [int] NOT NULL,

       [NAME\_VAL1] [varchar](50) NOT NULL,

       [DATE\_VAL1] [date] NULL,

       [NUMBER\_VAL1] [float] NULL,

 CONSTRAINT [PK\_Table\_2] PRIMARY KEY CLUSTERED

(

       [ID\_COL] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Table\_2]  WITH CHECK ADD  CONSTRAINT [chkRowIDMax] CHECK  (([ID\_COL]>=(1)))

GO

ALTER TABLE [dbo].[Table\_2] CHECK CONSTRAINT [chkRowIDMax]

GO

CREATE TABLE [dbo].[Table\_3](

       [ID\_COL] [int] NOT NULL,

       [NAME\_VAL1] [varchar](50) NOT NULL,

       [DATE\_VAL1] [date] NULL,

       [NUMBER\_VAL1] [float] NULL,

       [ErrorRow] [xml] NULL,

       [ErrorMessage] [varchar](1000) NULL,

       [ErrorCode] [varchar](1000) NULL,

       [ErrorColumn] [varchar](1000) NULL,

 CONSTRAINT [PK\_Table\_3] PRIMARY KEY CLUSTERED

(

       [ID\_COL] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

CREATE TABLE [dbo].[Table\_4](

       [ID\_COL] [int] NOT NULL,

       [NAME\_VAL1] [varchar](50) NOT NULL,

       [DATE\_VAL1] [date] NULL,

       [NUMBER\_VAL1] [float] NULL,

       [ErrorRow] [xml] NULL,

       [ErrorMessage] [varchar](1000) NULL,

       [ErrorCode] [varchar](1000) NULL,

       [ErrorColumn] [varchar](1000) NULL,

 CONSTRAINT [PK\_Table\_4] PRIMARY KEY CLUSTERED

(

       [ID\_COL] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

CREATE TABLE [dbo].[Table\_5](

       [ID\_COL] [int] NOT NULL,

       [NAME\_VAL1] [varchar](50) NOT NULL,

       [DATE\_VAL1] [date] NULL,

       [NUMBER\_VAL1] [float] NULL,

       [ErrorRow] [xml] NULL,

       [ErrorMessage] [varchar](1000) NULL,

       [ErrorCode] [varchar](1000) NULL,

       [ErrorColumn] [varchar](1000) NULL,

 CONSTRAINT [PK\_Table\_5] PRIMARY KEY CLUSTERED

(

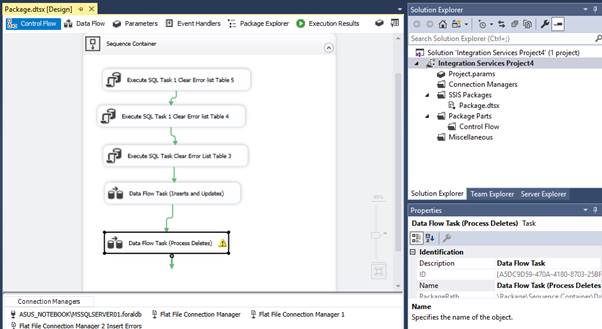
       [ID\_COL] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

Control Flow



i.e. Three upfront table truncate calls to clear some error logging tables (Table\_3, table\_4, table\_5)

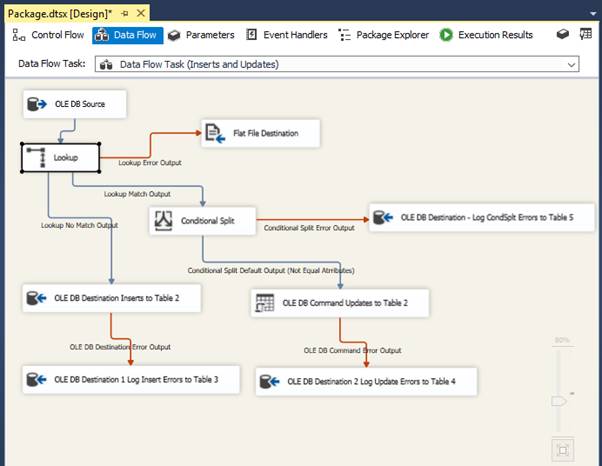
e.g. commands like:

truncate TABLE [foraldb].[dbo].[Table\_5];

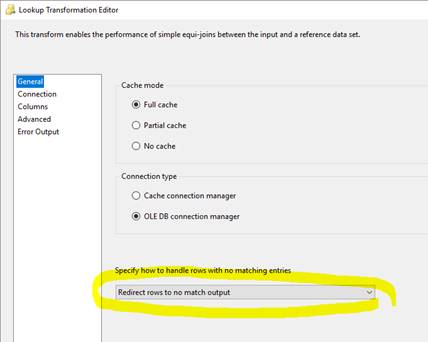
Then a Data Flow task to process Inserts and Updates

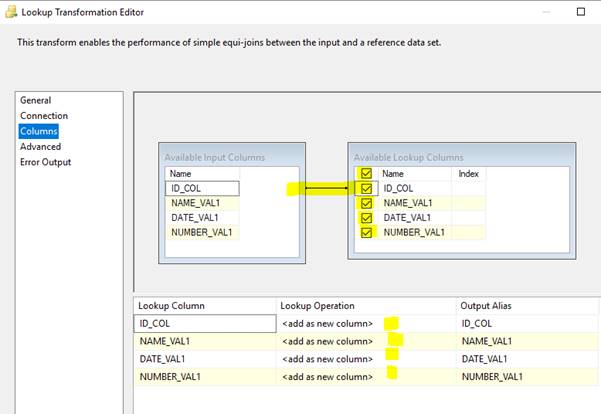
Then a Data Flow task to process any required deletions.

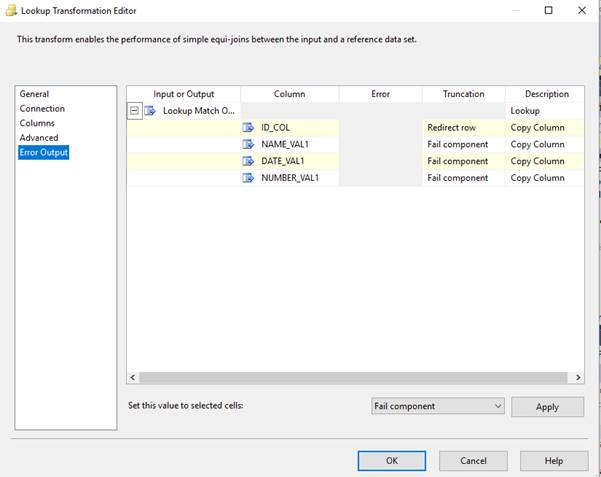
Data Flow Task (Inserts and Updates)



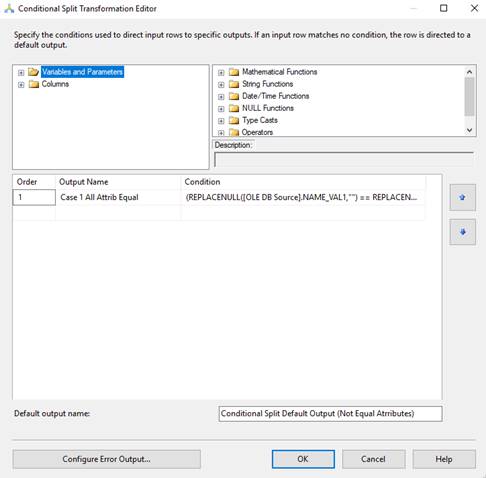
Settings for the Lookup component:







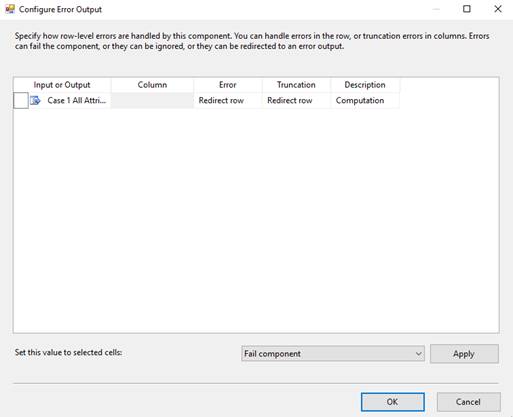
Settings for the Conditional Split component:



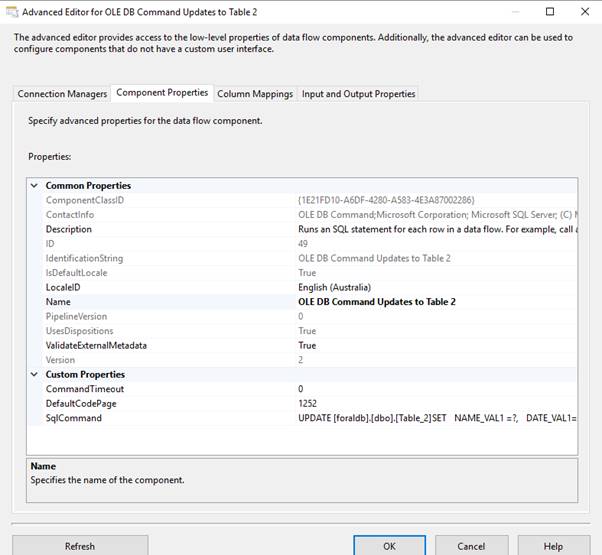
The condition text here is this:

 (I need to improve this to better handle nulls and not use a dummy hard coded number value … or date value ..) :

(REPLACENULL([OLE DB Source].NAME\_VAL1,"") == REPLACENULL(Lookup.NAME\_VAL1,"")) && (REPLACENULL([OLE DB Source].DATE\_VAL1,"1901-01-01") == REPLACENULL(Lookup.DATE\_VAL1,"1901-01-01")) && (REPLACENULL([OLE DB Source].NUMBER\_VAL1,-919.9) == REPLACENULL(Lookup.NUMBER\_VAL1,-919.9))



Settings for the OLE DB Command component (for the updates)



The SqlCommand here is:

UPDATE [foraldb].[dbo].[Table\_2]

SET

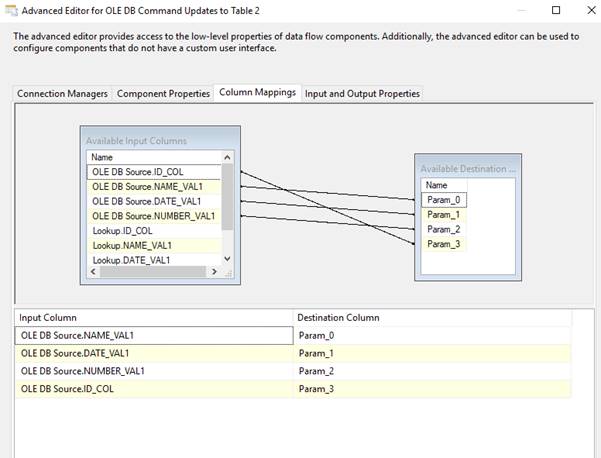
  NAME\_VAL1 =?,

  DATE\_VAL1=?,

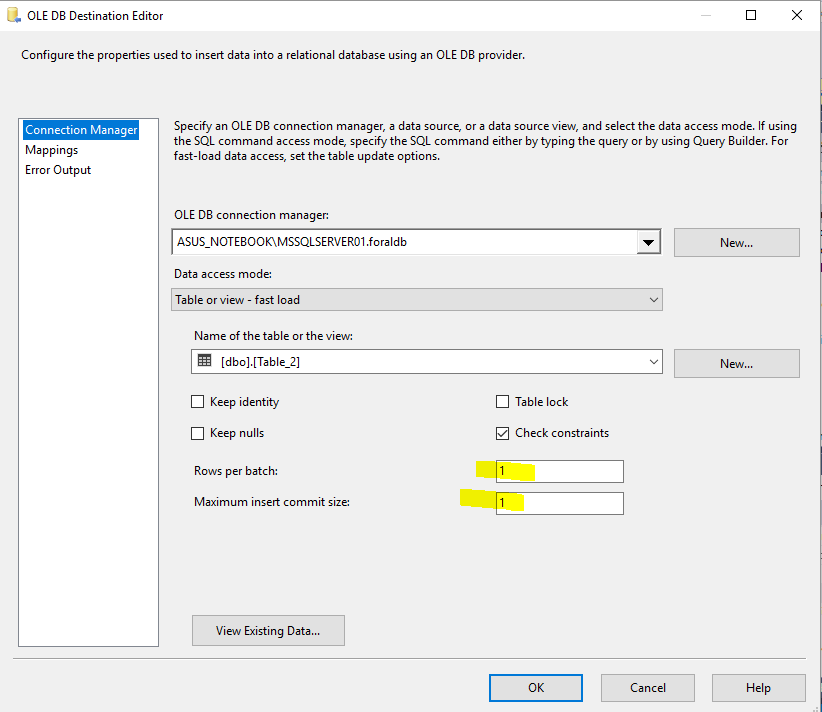
  NUMBER\_VAL1=?

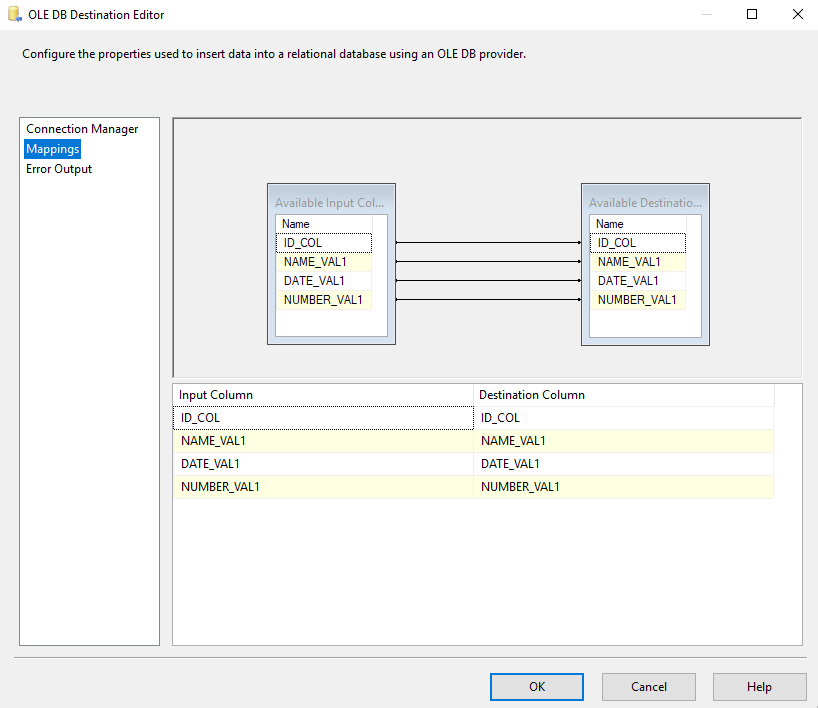
WHERE

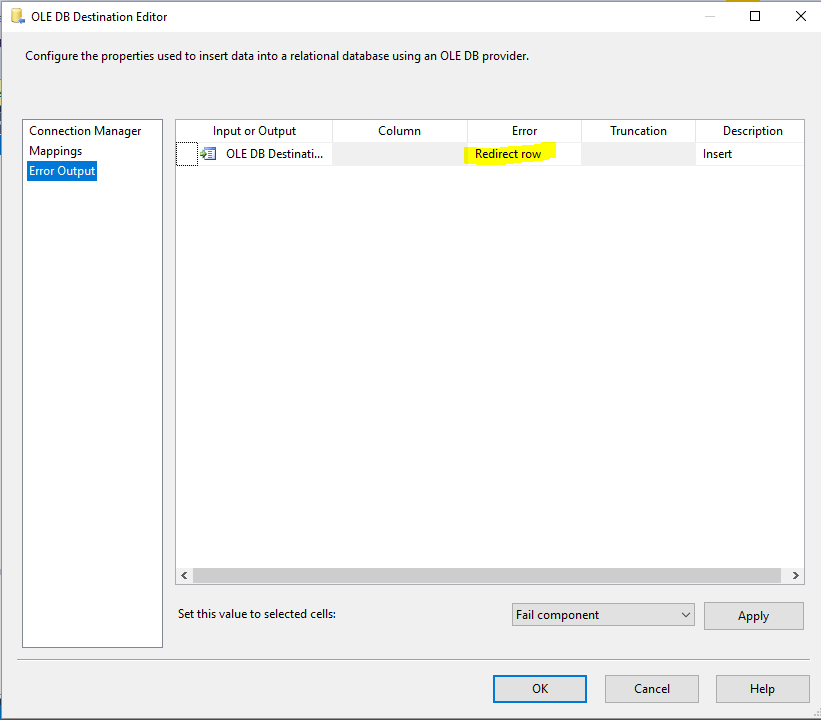
ID\_COL = ?



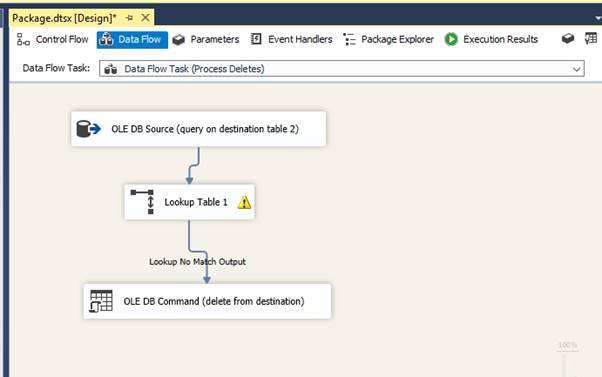
Some settings for the OLE DB Destination component are:



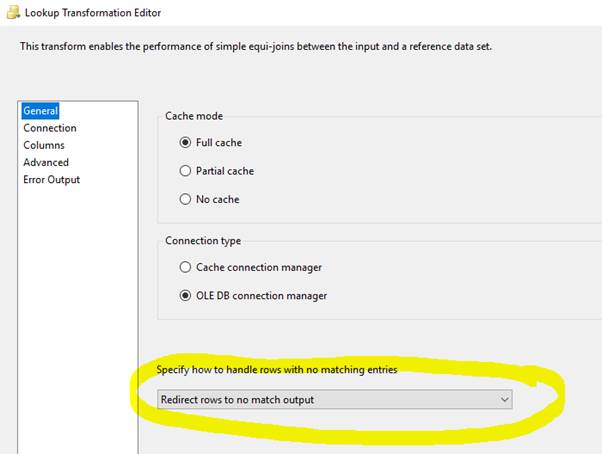


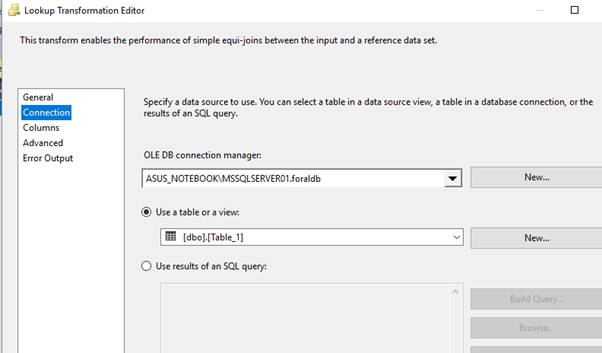


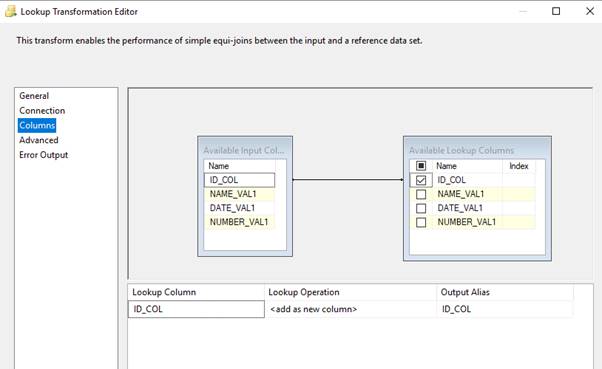
Data Flow Task (Process Deletes)

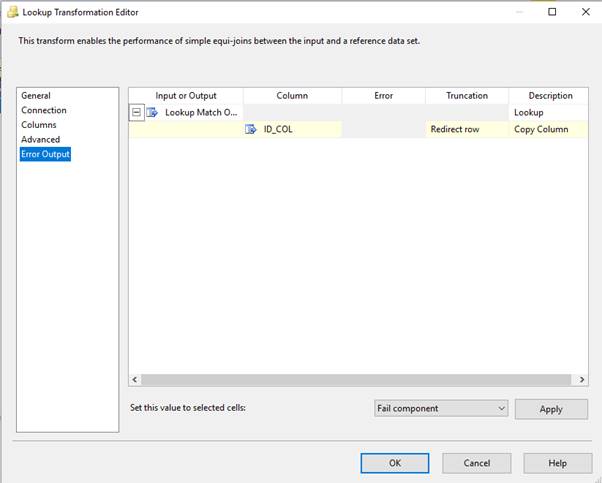


The lookup component settings here are:

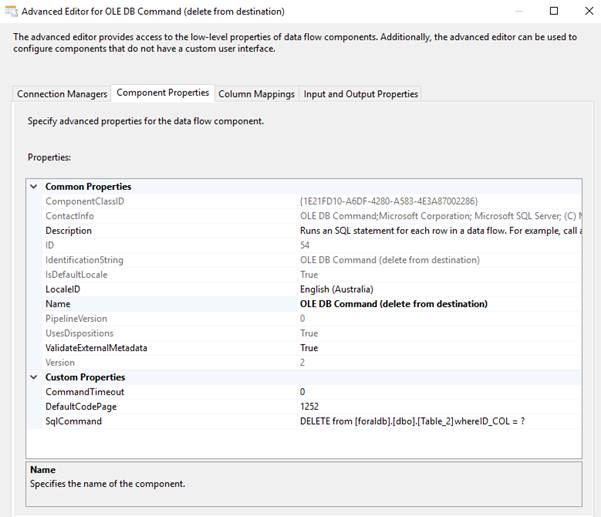








The settings for the OLE DB Command to process the delete command(s) are:

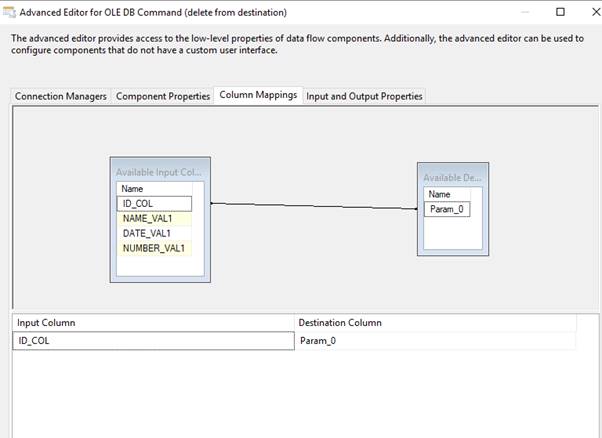


The SqlCommand is:

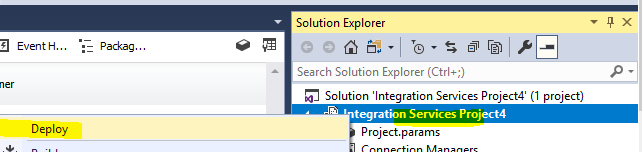
DELETE from [foraldb].[dbo].[Table\_2]

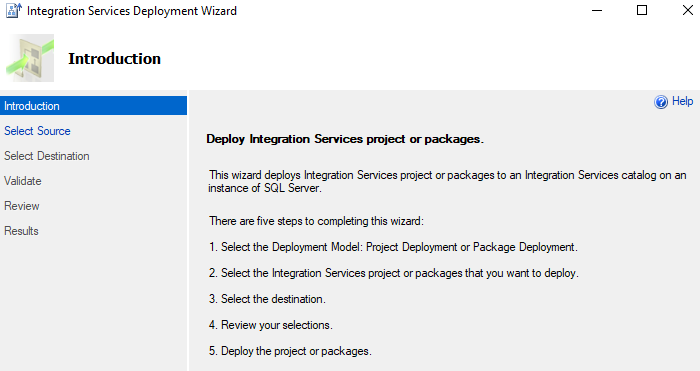
where

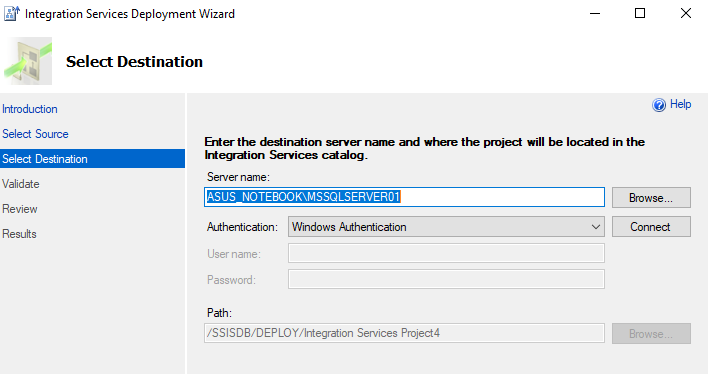
ID\_COL = ?

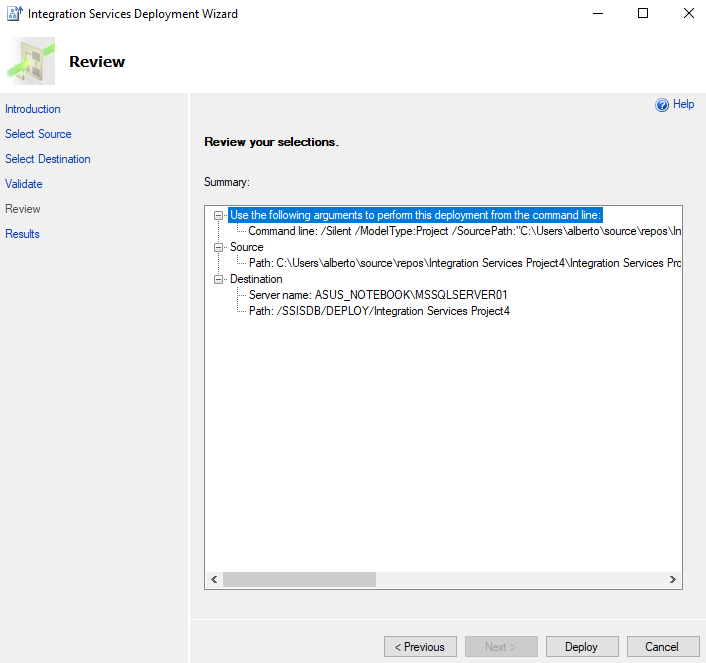


Note: When completed and tested, the process can be deployed to the SQL\*Server database.

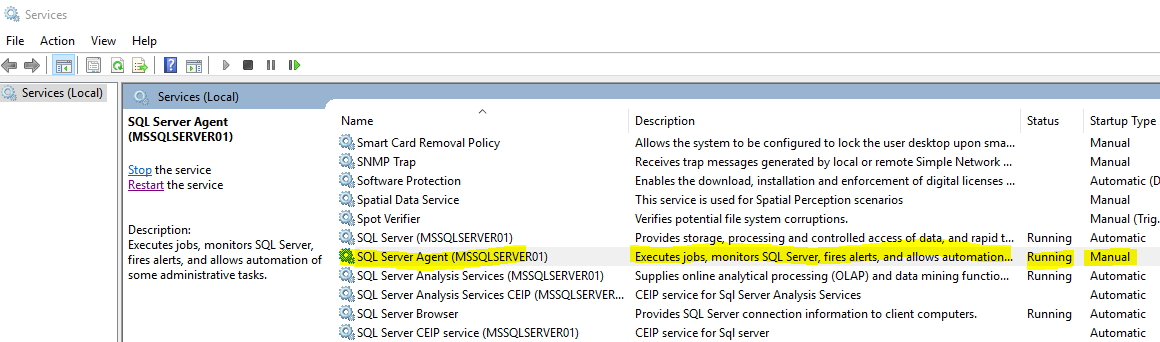








Start the SQL Server Agent service (service is set to manual startup at install time ..)



Create a job in SSMS tool (by right clicking Jobs heading and choosing new) ..  and schedule it (e.g. daily, weekly etc ..)  .

Can use “Job Activity Monitor” to look at process history as per below picture:

