Bonus Chapter B

Using SSIS as a Data Service

In earlier chapters, you saw how to use SSIS in many different ways. The script task and script component, in particular, make it easy to extend your use of SSIS. In this chapter, you will learn one final technique: how to deliver the results of data integration directly to a report. This is not strictly a scripting technique, but it does offer new opportunities for SSIS developers. This chapter assumes some familiarity with *reporting services*, especially how to create report projects and reports. For a good introduction, see *The Rational Guide to SQL Server Reporting Services*, in this series.

Traditional ETL or integration processes deliver data from databases, files, and applications. The processes must often transform the data from the sources. Data delivery is generally to a database that users can query and report on. As you have seen, SSIS scripting enables many you to source and transform the data in unique ways. With SSIS, you can also move beyond standard ETL when delivering data. In fact, you can drive a report from a simple or complex dataflow, without loading a database or other data store.

Why would you want to load a report directly from SSIS? When would you not want use a database for reporting? The answers to these questions really concern the nature of data integration today. Business data systems are becoming ever more dynamic. In many cases, data may be coming from the Web, or over RSS feeds, and users need quick access to this data. However, the raw data is often incomplete, or of poor quality. In such cases, users still need the data quickly, but the process must clean and transform the data first.

Another interesting use of this technique is to create a report over data that reporting Services cannot use alone. In fact, in the example in this chapter, you will create a report over the unusual source data from Chapter 9.

TechTip:

It is good to remember that with SSIS you do not need to choose between dynamically loading a report and using a database. SSIS enables parallel paths in the data flow and it is quite possible to load a report dynamically while also loading the data into a database.

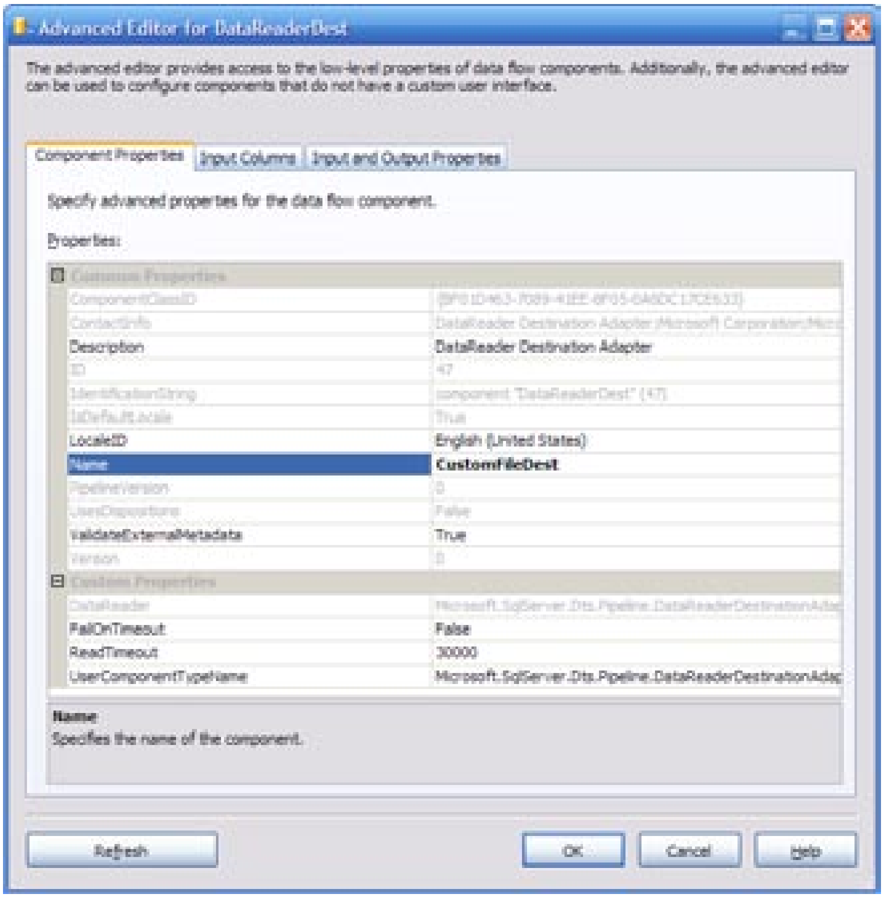
Creating an SSIS package to Write to a Report

The component that makes these interesting scenarios possible is the *DataReader Destination*. This component exposes the data arriving at its input as an ADO.NET DataReader. ADO.NET clients such as reporting services, which know how to execute SSIS packages, can connect to this DataReader and read its data. They do this by executing the package in the background.

In this example, you will create a very simple report using the non-standard source data in Chapter 9. Use the package you created in Chapter 9 for Listing 9.5, or go through that chapter again until you have created and tested that package.

Follow these steps to add a DataReader Destination.

1. Open the package you created for Chapter 9, Listing 9.5.
2. Delete the **Row Count** component.
3. Drag a **DataReader Destination** from the **Data Flow Destinations** tab of the toolbox to the **Data Flow** design surface.
4. Connect the output from your source **Script Component** to the **DataReader Destination**.
5. Double-click the **DataReader Destination** shape on the design surface to open the **Advanced Editor for DataReaderDest** dialog as shown in Figure B.1.
6. On the **Component Properties** page of the editor, rename the destination to an easy-to-remember name, such as `CustomFileDest`.


Advanced Editor for DataReaderDest

The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Component Properties | Input Columns | Input and Output Properties

Specify advanced properties for the data flow component.

Properties:

Common Properties	
ComponentClassID	{BF01D463-7089-41EE-8F05-0A6DC17CE633}
ContactInfo	DataReader Destination Adapter;Microsoft Corporation;Micr
Description	DataReader Destination Adapter
ID	47
IdentificationString	component "DataReaderDest" (47)
IsDefaultLocale	True
LocaleID	English (United States)
Name	CustomFileDest
PipelineVersion	0
UsesDispositions	False
ValidateExternalMetadata	True
Version	0
Custom Properties	
DataReader	Microsoft.SqlServer.Dts.Pipeline.DataReaderDestinationAdap
FailOnTimeout	False
ReadTimeout	30000
UserComponentTypeName	Microsoft.SqlServer.Dts.Pipeline.DataReaderDestinationAdap

Name
Specifies the name of the component.

Figure B.1: DataReaderDestination Editor.

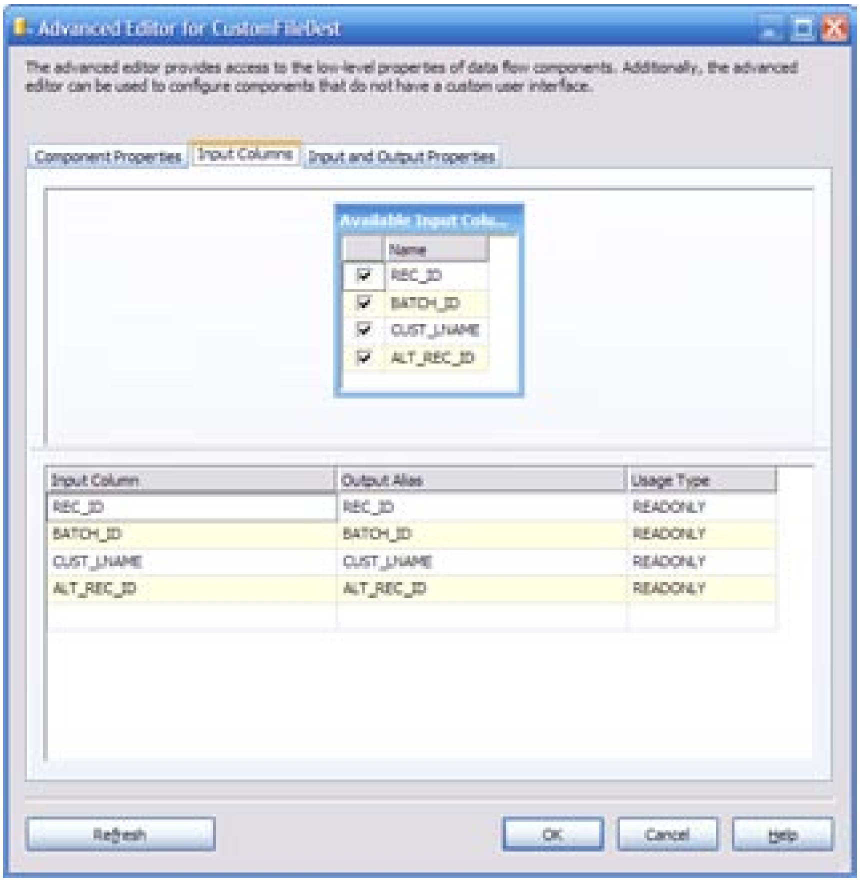
It is important to give your DataReaderDestination useful names. Later on, you will need to use these names when creating your report. It is handy to give them a name you can remember!

Next, you should select the columns you wish to see in your report. Often you will want to see all the columns. However, if you need to see only a subset, you can select them here. Follow these steps:

1. Navigate to the **Input Columns** tab of the editor.

2. Using the check boxes, select the columns you wish to use in your report. You can see this step in Figure B.2.

3. When you are ready, click the **OK** button to close the editor.



Advanced Editor for CustomFileDest

The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

	Name
✓	REC_ID
✓	BATCH_ID
✓	CUST_LNAME
✓	ALT_REC_ID

Input Column	Output Alias	Usage Type
REC_ID	REC_ID	READONLY
BATCH_ID	BATCH_ID	READONLY
CUST_LNAME	CUST_LNAME	READONLY
ALT_REC_ID	ALT_REC_ID	READONLY

Figure B.2: Selecting Columns in the DataReaderDestination.

At this point, you can run the package. However, when you do, there is not much to see. The script source will deliver rows to the DataReader destination. The shapes in the data flow designer will remain yellow for some time. This indicates that they are still in progress. In fact, they will remain in this state for around 30 seconds before turning green to show that they have completed. Why 30 seconds? Look again at Figure B.1. The **ReadTimeout** property for the destination is 30000 milliseconds. At the end of this time, if no client is reading the destination, the destination completes. Note that **FailOnTimeout** is **False** by default, so the destination does not fail in these cases.

Having created your package with a DataReaderDestination, you should save it with a useful name, according to the following steps. Again, this will be important when you come to create your report.

1. From the main menu of the Designer, choose the **File⇨ Save As** option for the package.

2. Save the package with a useful name, such as `CustomSource.dtsx` to a useful folder such as `C:\Samples`.

TechTip:

When you save your package with a new name, the designer will prompt you to rename the package object too. You should nearly always choose to do so. The *.dtsx file you save in Visual Studio is simply a special XML file. You can rename it using Visual Studio or the file system. However, the file contains the definition of an SSIS package. When the designer first creates it, the package object has the same name as the file. However, if you only rename the file, you will break that relationship. You could end up with a file called Package1.dtsx containing an object named Package2. For this reason, it is best to rename files using the designer, or the DTUtil utility that ships with SSIS.

Creating a Report to Read from an SSIS Package

Now it is time to create a report that will use the SSIS package. First, you must create a *report Project* in the designer. Follow these steps to add a data source:

1. Choose the **File⇨ New⇨ Project** menu item.

2. In the **New Project** dialog, select the **Report Server Project**. You can choose to create the project in the same solution you are currently using. Alternatively, you can create it in a new solution.

3. A new **Report Project** will appear in the **Solution Explorer**.

Configuring an SSIS Data Source for Your Report

Having created your project, the first step is to add a *data source* for your report to use. There is a special SSIS data source type for this purpose. Data sources need a connection string. For SSIS sources, the connection string is composed of the command-line switches needed to execute the package. In this example, you will simply use the **-f** command line switch, to execute the package from a file. However, you can also use other switches, including the ability to configure the package, and to set the values of variables within the package, and so on. For more information on command line switches, type `dtexec /?` at the command line. Follow these steps to add a data source:

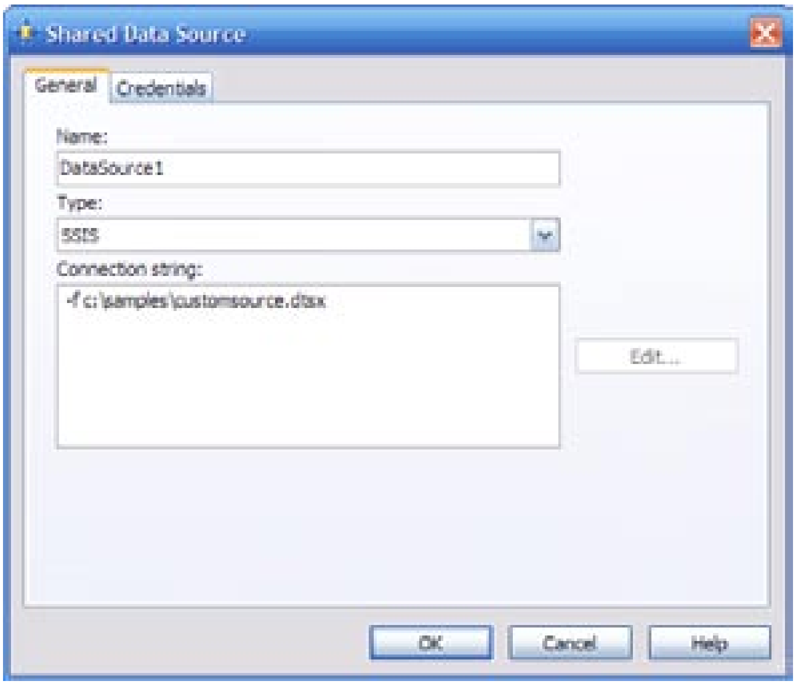
1. Right-click on the **Shared Data Sources** folder of your report project.

2. Select **Add New Data Source** from the contextual menu.

3. In the **Shared Data Source** dialog box, select **SSIS** as the data source type.

4. Type the following as your connection string:
   ```
   -f c:\samples\customsource.dtsx
   ```
   This tells SSIS to execute the package from a file, and gives the path to the file. There is no need to set **Credentials** for an SSIS data source.

5. Click the **OK** button to create your data source. See Figure B.3, which shows the completed **Shared Data Source** dialog box.



Name:
DataSource1
Type:
SSIS
Connection string:
-f c:\samples\customsource.dtsx

Figure B.3: Configuring a Data Source.

Creating the Report

Now that the data source is ready, you can build a report to use the data. Follow these steps:

1. Right-click on the **Reports** folder of your report project.
2. Select **Add New Report** from the contextual menu.
3. The **Report Wizard** will start.
4. Click past the **Welcome** page to **Select the Data Source**.
5. If you only have one data source in your project, the dialog will have selected it automatically. If not, select the appropriate **Shared Data Source** from the drop-down list.
6. Click the **Next** button to navigate to the **Design Query** page.

7. SSIS does not have a query builder. The query for an SSIS package is the name (case-sensitive) of the **DataReader Destination**. Now you know why it was important to give that component an easy-to-remember name! In this case, we named the object CustomFileDest, so enter that in the **Query String** pane.

8. Click the **Next** button. The wizard will look for the data source package. If it can find it, it checks for a Data Flow with the correct DataReader Destination in it. If all is good, then the next page is **Select the Report Type**. You can select any type of report. Click the **Next** button when you are ready.

9. On the **Design the Table** screen, you can choose which columns to use for **Pages**, **Groups** and **Details** in the report. This works for an SSIS Data Flow just as if you were building a report over a database table! Figure B.4 shows this screen in use for our current package.

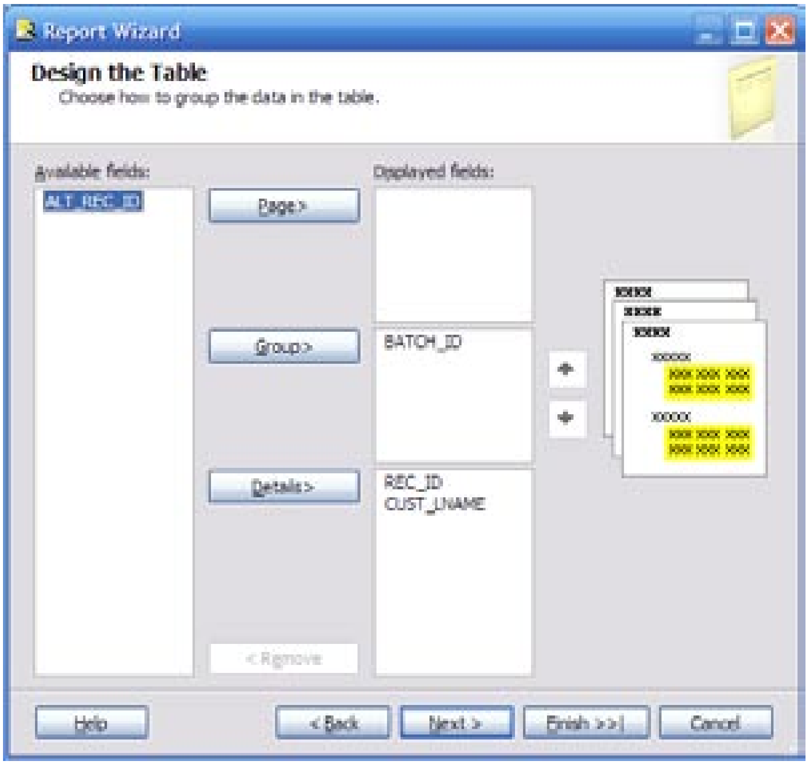
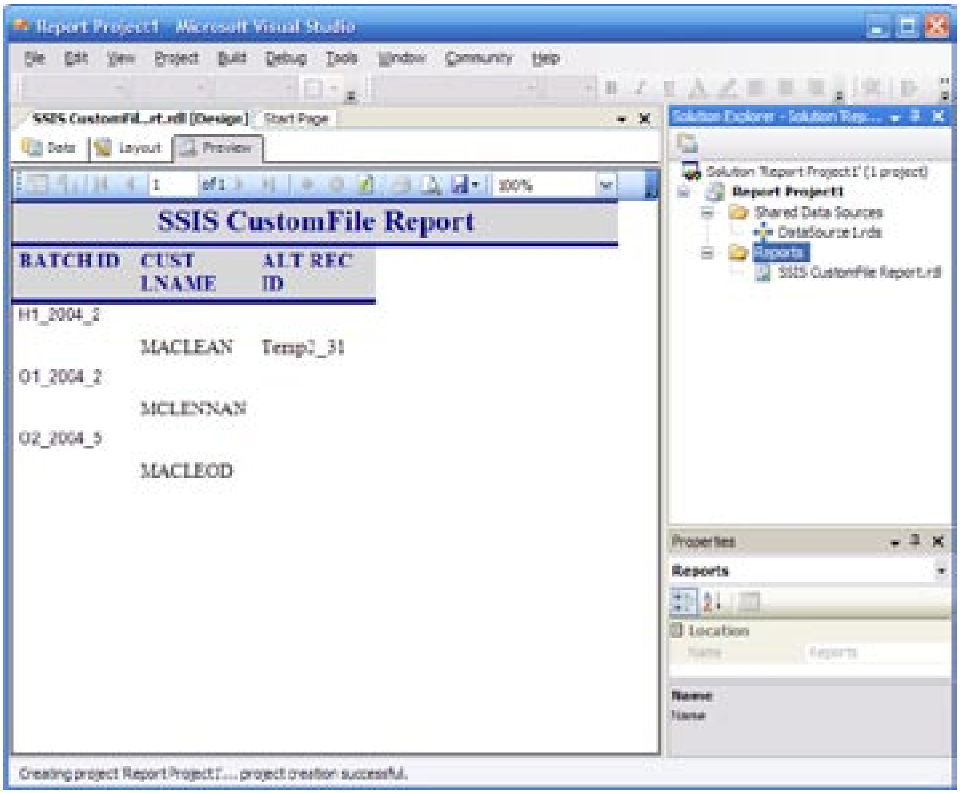

Report Wizard
Design the Table
Choose how to group the data in the table.
Available fields: ALT_REC_ID
Displayed fields:
Page: (none)
Group: BATCH_ID
Details: REC_ID, CUST_LNAME

Figure B.4: Designing a Report over an SSIS Data Source.

Note that the columns available here are the columns that you selected in the **DataReader Destination Editor**.

You can complete the report layout as you like using the remaining report wizard screens. Finally, give the report a name, and finish the wizard. Now you can preview your report, as shown in Figure B.5.


Report Project1 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Window Community Help
SSIS CustomFil...rt.rdl [Design] Start Page
Data Layout Preview
Solution Explorer - Solution 'Rep...
Solution 'Report Project1' (1 project)
Report Project1
Shared Data Sources
DataSource1.rds
Reports
SSIS CustomFile Report.rdl
Properties
Reports
Location
Name Reports
Name
Name

SSIS CustomFile Report

BATCH ID	CUST LNAME	ALT REC ID
H1_2004_2		
	MACLEAN	Temp2_31
Q1_2004_2		
	MCLENNAN	
Q2_2004_5		
	MACLEOD	

Creating project 'Report Project1'... project creation successful.

Figure B.5: Report Using an SSIS Package as a Data Source.

Reports built in this way execute the SSIS package in the background, whenever you refresh them.

If you deploy the report, it must still be able to find the Package. For this reason, you may want to ensure that the package is available on a network share. Also, ensure that the report's connection string can point to this location from the deployment report server.

Summary

In this chapter, you have seen how to use SSIS as a data source for a report. This technique, although very simple, is extremely powerful. It brings dynamic, near real-time Business Intelligence within the reach of all users. When used with SSIS scripting, it enables reporting services users to access live data with almost limitless transformation capabilities.