QlikView Expressor Tutorial Introductory

Contents

Tutorial 0

You may already have downloaded and installed QlikView Expressor Studio. If so, you only need to carry out the last three bullets on the following list.

Setup is quite minimal:

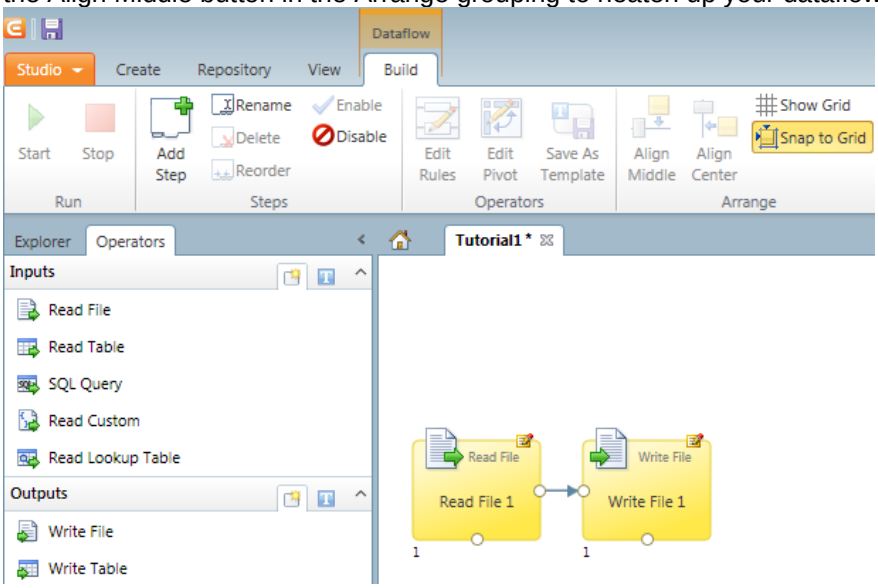
- Your computer must be using:
  - Windows XP, service pack 3, or
  - Windows 7, Professional or Enterprise, 32 or 64 bit
- Install QlikView Expressor Studio.
  - Download the software from the URL http://www.qlikview.com/us/explore/experience/expressor-download
  - The user installing Expressor Studio must be an administrator on the target computer.
- Create the directory C:\data.
- Download the PDF and ZIP files associated with this page to obtain the manual and data files used in this tutorial.
- Extract the ZIP archive file into the directory C:\data.

Tutorial 1

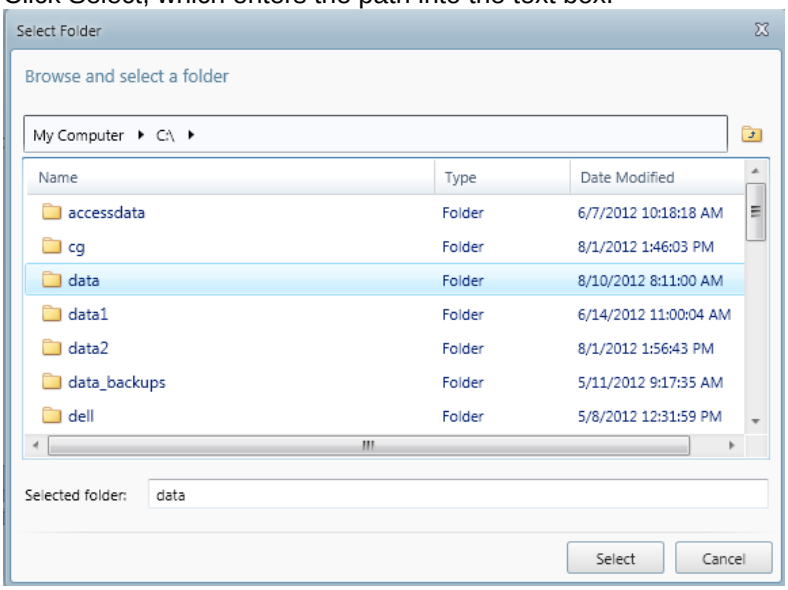
Every development tool presents a Hello World type application as a first example, so here goes: read a file, write a file. With QlikView Expressor Studio, creating this application is quite simple; the context sensitive help will guide you through each step of the process.

Step-By-Step Procedure

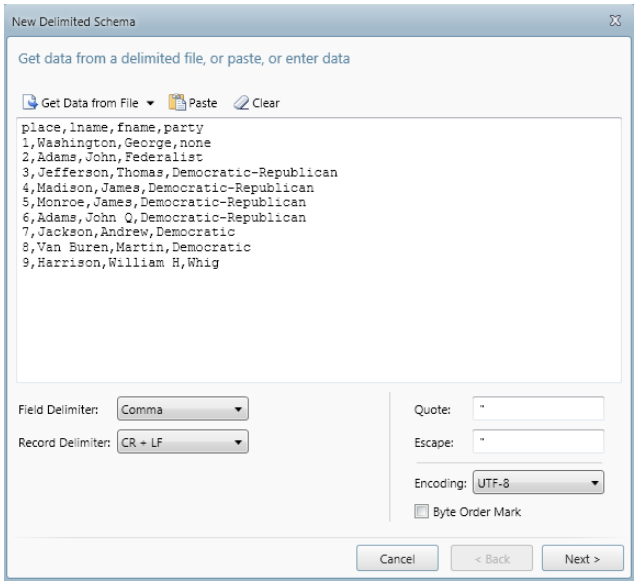
1. Use the Windows Start menu entry to open **Expressor Studio**. If this is the first time you have used Studio, you will need click the **New Workspace…** link, select **Standalone Workspace**, enter the name for the new workspace, and click **Create**.
    a. If desired, you can change the file system location for the workspace, but for these tutorials accept the offered default.
    b. A best practice is to enter a short description of the data integration applications you will develop within this workspace.
    c. Close the yellow 'Enable extensions for this workspace' bar. Extensions are not needed for this tutorial.
2. Within the **Getting Started** panel, click on the New Project link.
    a. Alternatively, click Project in the New Projects grouping on the Create tab of the ribbon bar.
    b. This will open the new Project window. Name the project Tutorial1, provide a brief description, and click Create.
    c. The project will be listed within the Explorer panel.
3. Click Dataflow in the New Artifacts grouping on the Create tab of the ribbon bar.
    a. Alternatively, highlight Tutorial1.0 in the Explorer tab, right click and select New… from the pop up menu.
    b. This will open the new Dataflow window. Name the dataflow Tutorial1, provide a brief description, and click Create.
    c. In the Explorer panel, the dataflow will be listed within the Dataflows subdirectory under the Tutorial1.0 project.
4. The Operators panel will come forward. Expand the Inputs and Outputs panes.
    a. Drag a Read File input operator and a Write File output operator onto the dataflow canvas.
    b. Use your mouse to connect the shapes then drag a box around both operators and click the Align Middle button in the Arrange grouping to neaten up your dataflow.



5. Select the Read File operator and note that the Properties panel displays context sensitive help and the control through which you configure this operator. The first entry you must make is to select a Connection, the artifact that tells the operator where the input file is located.
6. Click on the Action button (to the right of the Connection drop down control) and select New File Connection… from the pop up menu to open the New File Connection wizard.
    a. Click on the opened directory icon and browse to the file system location of the input file (for this example, C:\data).
    b. Click Select, which enters the path into the text box.

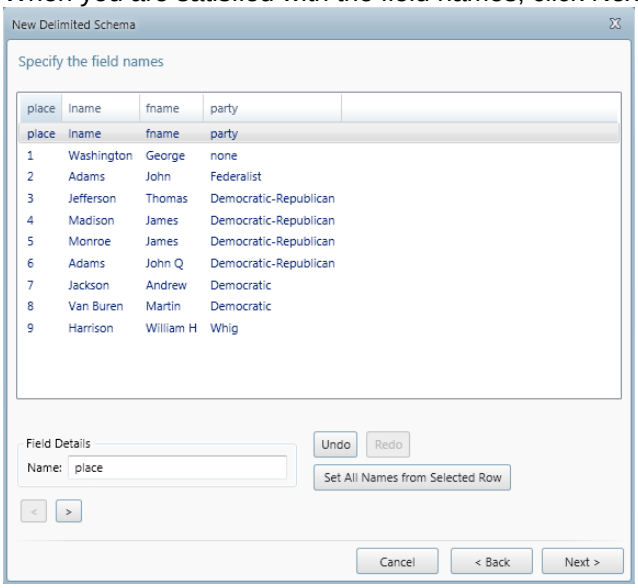


    c. Click Next, and in the following screen give the connection a meaningful name (for example, FileConnection), confirm that the connection artifact will be saved within the Tutorial1.0 project, and click Finish.
        i. Note that in the Explorer panel, the connection is listed within the Connections subdirectory under the Tutorial1.0 project directory.
7. In the Read File operator's Properties panel, note that your connection appears in Connection drop down control.
8. Next click on the Action button (to the right of the Schema drop down control) and select New Delimited Schema… from the pop up menu to open the New Delimited Schema wizard..
    a. The schema created with this wizard describes the structure of data that will be read from a delimited file.
    b. There are three ways you can use this wizard.
        i. Click Get Data from File and browse to and select the input file `C:\data\presidents.dat`. Note that rather than browsing through the file system, you can simply click on your connection and select your source file. This will enter the first 10 lines of data into the text box control.
        ii. Paste one or more lines of the input file into the text box control.
        iii. Manually enter one or more lines of data similar to the data in the input file. A nice option when using this approach is to enter a header row that includes the desired name for each field in the record.



Use the first approach, and after populating the text box control, select the appropriate Field Delimiter (Comma) and Record Delimiter (CR+LF). Your choices must match the field and record delimiters used within the data file. Then click Next.

9. In the following screen, you want to give each field a meaningful name. If your data file includes a header row, or if you manually entered a header row into the text box control on the previous screen, you can easily transfer the field names by selecting the first row and clicking Set All Names from the Selected Row.
    a. If desired, you can manually enter each field name, or change a field name by selecting a field and making an entry in the Name control in the Field Details grouping.
    b. When you are satisfied with the field names, click Next.



10. In the final screen, give your schema a meaningful name (e.g., PresidentsFileSchema), confirm that the schema will be saved into the Tutorial1.0 project, and click Finish.
11. In the Read File operator's Properties panel, note that your schema appears in Schema drop down control and that entries also appear in the Type and Mapping dropdown controls.
    a. In this example, these are your only choices. In later tutorials, you will learn how to use the Type and Mapping options to provide enhanced functionality within the application.

12. To complete configuration of the Read File operator, enter the name of the input file into the File name control, leave the Quotes**,** Error handling and Show errors controls unchanged and, if the input file includes a header row (or rows), enter the number of rows to skip into the Skip rows text box (in this example, enter 1).
    a. In this example, the name of the input file is `presidents.dat` and one row should be skipped.
13. Now select the Write File operator. Select the same Connection, Schema, Type and Mapping entries as for the Read File operator.
    a. Enter a name for the output file into the File name control (e.g., `presidents_out.txt`), then select any entry in the Quotes control.
    b. If you want the output file to include a header row, select the Include header check box.
    c. If you want to append output to content that is already in the target file, select the Append to output check box.
    d. You may also select the property that appends a timestamp to the name of the output file.
14. Save the dataflow.
15. Run the dataflow by clicking the Start button in the Run grouping on the Dataflow > Build tab of the ribbon bar.
16. Use Windows Explorer to browse to the location (`C:\data`) of your output file and review its contents.

Note that the contents of the input file have been copied to the output file. To convince yourself that the input file's content has been processed by the data integration application, run the dataflow multiple times, enabling and disabling the output file's header row.

Tutorial 2

In Tutorial1, you developed an Expressor data integration application in which all artifacts were contained within a single project. An **Expressor project** may include many connection, schema, and dataflow artifacts, and can thereby support an extensive development effort. To provide for reuse of artifacts, you may also create artifacts within an Expressor library that can then be associated with one or more Expressor projects (or other Expressor libraries).

Step-By-Step Procedure

1. Working within the same workspace as Tutorial1, select the Create tab on the Ribbon Bar and click New Project.
    a. Create a new project named Tutorial2.
2. Next, click New Library, which opens the New Library window.
    a. Create a library named MyLibrary.
3. In the Explorer panel, expand the library MyLibrary.
    a. Right-click on the Connections subfolder and select New > File Connection from the popup menu. This opens the New File Connection wizard you used in Tutorial1.
        i. As in Tutorial1, create the file connection to the C:\data directory. Name the connection FileConnection2.
    b. Right-click on the Schemas subfolder and select New > Delimited Schema > From Data Source from the popup menu. This opens the New Delimited Schema wizard you used in Tutorial1.
        i. As in Tutorial1, create the delimited schema. Name the schema PresidentsFileSchema_2.
4. In the Explorer panel, expand the project Tutorial2.0.
    a. Right-click on the Library References subfolder and select Library References... from the pop up menu. This opens the Library References window.
        i. The panel on the left lists the libraries within this workspace. Check the box next to the Expressor library, or libraries, you want to associate with the Tutorial2.0 project. Click OK.
5. Under the Tutorial2.0 project folder.
    a. Right-click on the Dataflows subfolder and select New… from the popup menu. Name the dataflow Tutorial2.
    b. As in Tutorial1, recreate the Read File, Write File data integration application. Note that you can select the connection and schema artifacts that are included within MyLibrary just as if they were included in the project itself.

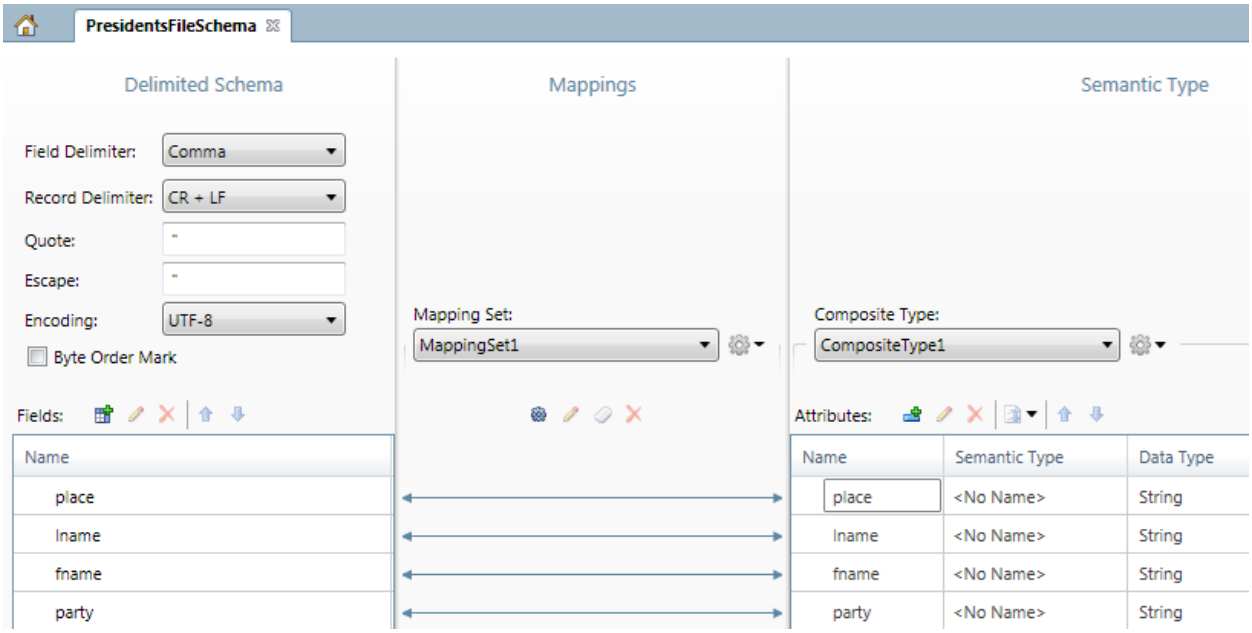
Artifacts may be moved or copied between Expressor projects and Expressor libraries. Select an artifact, right-click, and choose the desired action and target project or library.

Tutorial 3

In the preceding tutorials, you created a schema that described the data fields in a file. In completing the example, you did not review the contents of the schema but just used it, and its encapsulated default composite type and mapping, to configure the Read File and Write File operators. In this tutorial, you will take a closer look at the schema and begin to learn about semantic types and mappings.

Step-By-Step Procedure

Expand the Tutorial1.0 project folder in the Explorer panel and then expand the Schemas sub-folder. Double-click on the schema, or right-click on the entry and select Open from the popup menu, to open it in the Schema Editor.



The schema artifact actually includes three types of information.

1. The left-hand column, Delimited Schema, displays the details of each field in the data resource. In this example, the entries are the field names in the file's header row.
    a. The Field Delimiter drop down control shows the character used as the delimiter between fields in each record and the Record Delimiter drop down control shows the control characters used to demarcate records. These were the values you specified when you created the delimited schema.
2. The right-hand column, Semantic Type, displays the details of each attribute in the composite type that corresponds to a field in the data record.
    a. The Composite Type drop down control, which displays the default composite type, allows you to select an alternative representation for the record by choosing a different composite type. Currently, there are no other composite types to select, but you will change this situation in a following tutorial.
3. The center column, Mappings, displays the mappings between the schema fields and the composite type attributes.

What does all this mean?

- The purpose of the schema artifact is to associate fields in the data resource to their corresponding representations – attributes – within the Expressor data integration application.
- When you created the schema, a default composite type and a default mapping set where also created.
- Since the schema wizard had no other guidance than the names of the fields in the data file, it gives each attribute the same name as its corresponding field,
- When the schema is used with the Read File operator, the value in the mapping entry means that "the value in the input record data field named 'place' will be used to initialize the attribute 'place' in the composite type." When the schema is used with the Write File operator, the value in the mapping entry means that "the value in the composite type attribute name 'place' in the composite type will be used to initialize the output record data field named 'place'." The mapping format describes the structure of the data in the input or output record data field.

Tutorial 4

In the Schema Editor, the information displayed in the left-hand panel is derived from the data resource's metadata. For a file data resource, each field is a string so the only metadata of substance is the name of the field. The metadata that describes a database table is much richer and includes the column type and size as well as its name. You will use the field detail information when you extend the contents of the schema file by adding mappings to alternative composite types.

In the preceding tutorial, the default composite type generated when you first created the schema contains four String attributes since each field in the file resource is a string type. This may be fine for your intended application, but it could present some problems. For example, in the listing of presidents, the place field represents the president's chronological position in the sequence of individuals who have held this office. If in your application you want to sort the records based on the value in this field, the sort order will correspond to a character sort, where presidents 10 through 19 will come before president 2, and not a numeric sort. This is probably not your intended outcome. This issue can be resolved by using the schema to change the attribute to a numeric type and using the mapping to enforce any desired formatting directives.

What you need to do is to either alter the default composite type or create a new composite type so that the Data Type of this attribute is Integer or some other numeric type.

- A best practice is to create a new composite type and mapping set rather than modifying the default composite type and mapping. This insures that you can always review the original schema content created by the Expressor wizards.

As with other well designed user interfaces, there are alternative approaches to creating a composite type and integrating it into the schema. With Expressor Studio, you can either use the Type Editor to create a new type and then add this type into the schema, or you can work within the Schema Editor, copying and modifying the default composite type. Each approach has advantages and one or the other may be more suitable in different situations, but ultimately the outcome will be the same. Let's first try the copy/modify scenario.

Step-By-Step Procedure

1. Open the PresidentsFileSchema in the Schema Editor.
2. Select the Schema > Edit tab on the ribbon bar.
3. Click Assign > Add To Schema > New (local)… in the Semantic Type grouping, which immediately creates a second composite type named CompositeType2. Click OK then click on the Composite Type drop down control to convince yourself that the original default composite type CompositeType1 is still available.
4. Be certain that the CompositeType2 composite type is selected, then in the Name column highlight the place attribute and click Edit > Attribute in the Editing grouping (or on the pencil icon in just above the listing of attributes), which opens the Edit Attribute window.
5. From the Data type drop down control, select Integer and then click OK.
   Note that the Data Type specification for this attribute changed from String to Integer.
6. Be certain that the CompositeType2 composite type is selected, then in the Mappings column highlight the place mapping and click Edit > Mapping Formatting in the Editing grouping (or click the pencil icon above the mappings), which opens the Edit Mapping Formatting window.
7. Since this is an integer value, on the Digits tab, select the Specify fractional digits radio button and enter zero for the minimum and maximum digits after the decimal. Click OK.
8. Save the modified schema.

With this new composite type, the value in the place field will be managed as an Integer value within the Expressor dataflow. You will only need to specify this mapping set/composite type pairing when setting the Type and Mappin**g** options for the Read File and Write File motors.

1. Open the Tutorial1 dataflow, select each operator and choose the CompositeType2 and MappingSet**1** entries from the Type and Mapping drop down lists.
   Note that the Write File operator turns yellow after you change the mapping in the first shape. To understand why, read the summary on the Messages tab of the Status panel.
2. Save the modified dataflow.
3. Run the dataflow and review the output.

Looks fine, right? But how do you know that within the Expressor dataflow the data type of this attribute is Integer and not String? Let's place a Transform operator between the existing operators and see what can be learned.

1. Select and delete the link between the operators then increase their separation to allow space for another shape.
2. Click the Operators panel and drag-and-drop a Transform operator (in the Transformers pane) onto the dataflow.
3. Connect the Read File operator to the input (left side) Transform operator, and the Transform operator output (right side) to the Write File operator.
   The Transform operator will remain yellow until you review/add scripting code.
4. Select the Transform operator and in its Properties panel click the Edit Rules link, which opens the Rules Editor. Alternative ways to open the Rules Editor are to select the operator and click Edit Rules in the Operators grouping of the Dataflow > Build tab of the ribbon bar or to double-click on the operator itself.
   a. Note that the coding panel is blank. Look at the listing of attributes in the right-hand panel and note the right facing arrow before each attribute's name. This arrow indicates that the attribute propagation feature has transparently copied the data in the incoming attributes into the outgoing attributes. You only need to provide code if you want to modify this direct transfer, which is not necessary in this example.
5. Carefully hold the cursor over each record field in the incoming record, the left-hand panel in the Transform Editor, and note the data type.
   a. Do the same thing for the outgoing record.

The types of each field correspond to the Data type assigned to the underlying attribute.

What significance is there to the name of the type and mapping options in the input and output operators? And where are these composite types stored?

With a quick glance you can discern that the mapping option is the name of a mapping set and the type is the name of a composite type, which you can view in the Schema Editor. If desired (not necessary in this tutorial), you can manage your mapping sets by selecting the Schema > Edit tab in the Schema Editor ribbon bar and using the buttons within the Mappings grouping. Similarly, for the Semantic Type, use the buttons within the Semantic Type grouping to manage your local composite types.

Mapping sets are always local to the schema; they are not a reusable artifact. Composite types can be either local to the schema or sharable across all schemas within a project or library. In this example you created a new local composite type, which was automatically named CompositeType2. To convert this type into a sharable composite type:

1. Open the Tutorial1 PresidentsFileSchema in the Schema Editor and from the Composite Type drop down list select the composite type (i.e., CompositeType2) you want to share.
2. Select the Schema > Edit tab on the ribbon bar and click the Share > Composite Type… button in the Semantic Type grouping.
3. In the Share Local Composite Type window, the project in which the type will be stored is preselected. Give the type a meaningful name, such as PresidentDetails, and click OK.

An entry for the sharable type appears under the Types sub-folder of the desired project (or library if you promoted the type from a schema in a library).

In this exercise, you used the Schema Editor to create a new composite type and mapping set within a schema and then converted the type to a reusable artifact. But you can work in the opposite direction: create a sharable type first then add it to the schema. This approach is described in the next tutorial.

Note that in your dataflow, the Read File and Write File operators have turned yellow indicating that they are not properly configured. You must now select PresidentDetails from the Type drop down control as the local type CompositeType2 no longer exists.

Tutorial 5

In the last tutorial you started to learn about Expressor semantic types. You opened a delimited schema, saw how fields are mapped to attributes, created a new local composite type, and then promoted the composite type so that it became a reusable artifact. In the future, you can use this composite type when creating a new mapping in another schema file.

In this tutorial, you will learn how to create a composite type using the Type Editor. Any types you create using this approach are created as reusable artifacts.

Before you start, let's review some terminology important to Expressor.

- There are two categories of semantic types: Atomic and Composite.
- A Composite Type is a Semantic Type that defines a simple or complex data structure. It contains a set of attributes, each of which is defined by an Atomic Type.
- An Atomic Type is a Semantic Type that defines a single cohesive unit of data that cannot be decomposed and accordingly is associated with a data type such as a string or integer. An Atomic Type is used to define the type of an attribute within a Composite Type.

In the previous tutorials, the discussion only dealt with composite types, but you now understand that each of the attributes was an atomic type. With this understanding, you are ready to create a composite type.

Think of a composite type as a description of a record as it moves through your data integration application. Immediately after a data record enters your application, its corresponding composite type describes the structure of the incoming record. In the previous example, the Read File operator retrieved four field records from the input file. The default composite type within the schema mapped each of these fields to string attributes, but in a second composite type that you added to the schema, you mapped one of the fields to an integer. Consequently, the record passed by the Read File operator to the next operator contained an integer field and three string fields.

It is very likely that your data integration application will manipulate the incoming data so that the structure of the record changes during the processing and the record emitted from the application will have a different structure than the original incoming record. To develop this type of application, you may choose to create composite types that describe the structure of the record as it passes through your application. Alternatively, you could simply specify the attributes comprising the record emitted from an operator locally within the operator itself.

As you create composite types, you may decide that some of the atomic types corresponding to the attributes would be useful when defining other composite types. Using the Type Editor, you can create reusable atomic types as well as reusable composite types.

Let's plan to modify the Tutorial1 dataflow from the last tutorial so that it applies a transformation to each record rather than just passing the records through to the output. Currently, each record includes the last name and first name of a president. You want the record emitted from the application to concatenate these two fields into a single field.

Your dataflow will include three operators: Read File, Transform, Write File. You already have a schema and mapping to use with the Read File operator. What else is needed?

- A composite type that describes the modified record's three fields: place, full name, and party.
- A schema that the Write File operator uses to emit the modified record.

After a little more thought, you decide that the attributes corresponding to the full name and place would be useful in other applications, so you decide to create atomic types that could be reused in other composite types.

Step-By-Step Procedure

1. On the Create tab of the ribbon bar, click the Type > Atomic Type button in the New Artifacts grouping.
2. Name the type place and select the Tutorial1.0 project as the storage location, then click Create.
3. An entry for the type appears under the Types sub-folder in the selected location and the Type Editor opens with the type selected.
4. From the Data type drop down list, select Integer then save the modified type definition.
5. Create a second atomic type named fullname, data type String. Save it to the same project.

Now you can create the composite type that encapsulates these atomic types.

1. On the Create tab of the ribbon bar, click the Type > Composite Type button.
2. Name the type PresidentName and select the same storage location (Tutorial1.0) as the atomic types, then click Create.
3. An entry for the type appears under the Types sub-folder in the selected location and the Type Editor opens with the type selected.
4. On the Type > Edit tab of the ribbon bar, click Add Attribute in the Add Items grouping. This opens the Add Attribute window. Give the attribute the name place.
5. Next, click the Actions button and select the Assign > Shared… menu item from the pop up menu.
6. In the Select Shared Atomic Type window, select the type place, click OK, then close the Add Attribute window.
7. Add a second attribute, named fullname, to this composite type and assign the atomic type fullname to this attribute.
8. Add a third attribute, string type, named party. Since the president's political party is not likely to be used in other data integration applications, you are satisfied that its Semantic Type is only local to this composite type.
9. Save the composite type.

Now that you have the necessary composite type, you can use it to generate a schema for the Write File operator, after which you will have all the artifacts you need to develop the application.

1. If necessary, open the PresidentName composite type in the Type Editor by double-clicking on its entry in the Explorer panel
2. Click New Schema > Delimited Schema from Type button in the Generate grouping on the Type > Edit tab of the ribbon bar.
3. As an alternative to steps 1 and 2, expand the Types sub-folder under theTutorial1.0 project, select the PresidentName composite type, right-click, and select the New Delimited Schema from Type… menu item.
4. Move through the New Delimited Schema from Type wizard.
5. You can accept the suggested name or enter a more descriptive one such as PresidentName_Delimited.
   Save the schema to the storage location (Tutorial1.0 project) that contains the types.
6. An entry will appear under the Schemas sub-folder.

Tutorial 6

Now that you have a composite type that describes the contents of your modified record, and a schema that will write this record to a delimited file, you are ready to reconfigure your dataflow to carry out the transformation.

In this tutorial, you will investigate alternative approaches.

- You will use the PresidentName composite type to describe the structure of the record emitted from a Transform operator.
- You will define the structure of the emitted record directly within the Transform operator.

Which approach you take in a specific dataflow really depends on how many data transformations you want to perform. The composite type approach is best when there are multiple transformations while the alternative approach is better when there are few transformations.

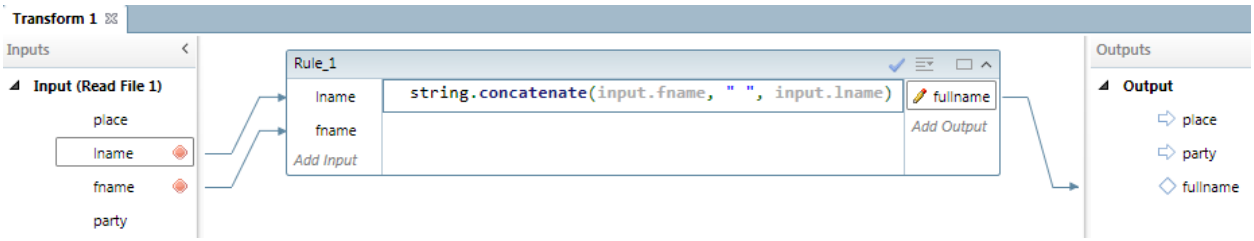
Step-By-Step Procedure

1. Working within the Tutorial1 project, create a new dataflow named Tutorial6.
2. Place a Read File operator onto the dataflow and in its Properties panel select PresidentsFileSchema, PresidentDetails type, and MappingSet1. Specify the file connection and enter `presidents.dat` as the source file. Be certain to configure the operator to skip one row.
3. Place a Transform operator onto the dataflow and connect to the Read File operator.
4. Select the Transform operator and open its Rules Editor.
5. When the Rules Editor opens, note that there is no coding in the panel. Also observe the listing of the four incoming attributes and the four, identically named outgoing attributes. The right-facing arrow before each outgoing attribute name indicates that Expressor's attribute propagation functionality will automatically transfer the incoming attribute value to an identically named attribute in the output.

Let's try the approach in which you describe the output record from within the Transform operator first. This approach utilizes the attribute propagation feature to its fullest extent.

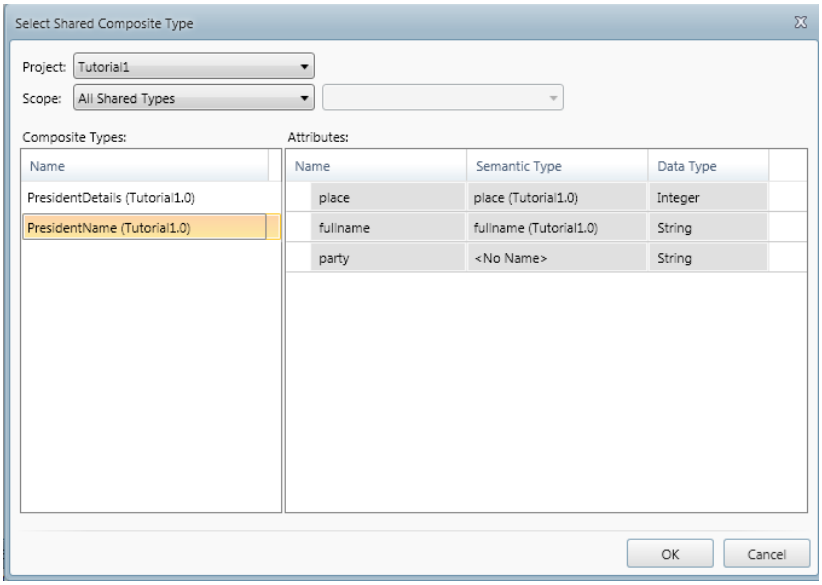
1. Press the Shift key and click on the lname and fname input attributes. In the ribbon bar, click the Block Inputs button in the Transfers grouping in the Home tab of the ribbon bar and notice that the attributes are removed from the output panel.
2. To add an attribute to the output record, click Add in the Output Attributes grouping in the ribbon bar. This open the Add Attribute window.
3. Create a string attribute named fullname and click OK. The right facing arrow icons to the left of the place and party output attributes indicates that these attributes will be transparently initialized through Expressor's attribute propagation feature. The diamond icon to the left of the fullname output attribute indicates that this attribute must be initialized within this operator

4. Select both the lname and fname incoming attributes and drag to the fullname outgoing attribute. Into the expression rule box, enter
`string.concatenate(input.fname, " ", input.lname).`



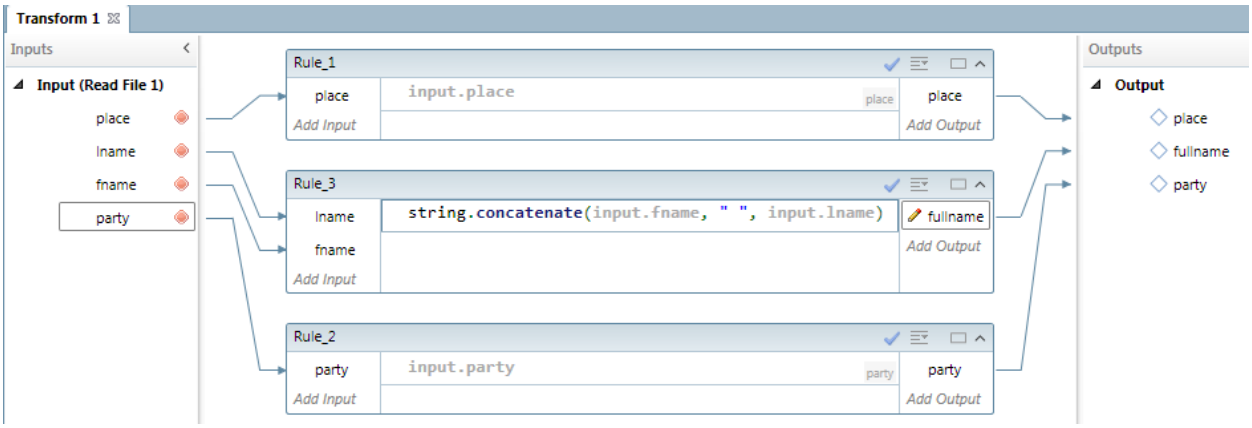
Now let's discuss the composite type approach. In this situation, you do not want to utilize (or fully utilize) the services of the attribute propagation feature. Rather, you want to assign values to each of the output attributes.

1. Press the Shift key and click on each of the input attributes. In the ribbon bar, click the Block Inputs button in the Transfers grouping in the Home tab of the ribbon bar and notice that the attributes are removed from the output panel. It's now up to you to define the structure of the output record.
2. Click Import in the Output Attributes grouping to open the Select Shared Composite Type window.
3. Select the PresidentName type and click OK.



4. In the Rules Editor, the three attributes from this composite type now represent the outgoing attributes. The diamond icon to the left of each output attribute indicates that this attribute must be initialized within this operator.
5. Using your mouse, drag a line from the incoming attribute place to the identically output attribute. Repeat for the attribute party. The necessary coding is automatically created for the place and party attributes.

6. Next, select both the lname and fname incoming attributes and drag to the fullname outgoing attribute. Into the expression rule box, enter
`string.concatenate(input.fname, " ", input.lname).`



Regardless of which approach you selected, you can now complete the dataflow by adding the Write File operator.

1. Place a Write File operator onto the dataflow and connect to the Transform operator.
2. In its Properties panel, choos the PresidentName_Delimited schema. This schema contains only one composite type and mapping set. Select the PresidentName type and MappingSet1 from the Type and Mapping drop down controls. Use the same file connection and enter a meaningful name for the output file.
3. Save and run the dataflow and observe the content of the output file.

Tutorial 7

Now that you have some experience accessing delimited files and working with atomic and composite types, schemas, and mapping sets, it's time to turn your attention to accessing database tables. The approach is very similar to what you did with delimited files except that the connection contains significantly more information and the metadata that describes the data resource is much richer.

The example in this tutorial will be built around the Microsoft SQL Server AdventureWorks database. If you have a different relational database management system or a different SQL Server database, it doesn't matter. The process is exactly the same.

NOTE: Tutorials 7, 8, and 9 read from database tables and write to files. If you do not have access to the source database, review these tutorials then move on to tutorial 10, which develops the same data integration application using file input.

Step-By-Step Procedure

Before starting, create a new standalone workspace.

1. In the New Workspace window, enter the name, workspace_2**,** accept the default location, enter a brief description, and click Create.
2. Create a new project named Tutorial7. Use either the New Project link in the Tasks panel or the New Project button on the Home tab of the ribbon bar.

Now you need to create a connection to the database. If your application involves moving data between databases, you will need to create multiple database connections. For the relational database management systems formally supported by Expressor Studio, the steps to create a connection are the same regardless of which database management system you are using. If you want to access an unsupported database management system, or an ODBC compliant data source such as Microsoft Access, you would use a different procedure that utilizes a pre-existing ODBC data source name.

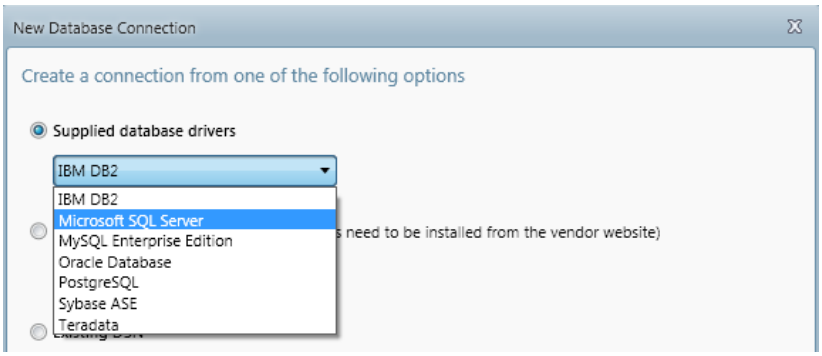
By now you know that there are multiple ways to start the process to create a connection.

- On the Create tab of the ribbon bar, click Connection > Database Connection.
- In the Explorer panel, right-click on the project and select New Connection > Database Connection from the popup menu.
- In the Explorer panel, expand the project, right-click on the Connections sub-folder and select New > Database Connection from the popup menu.

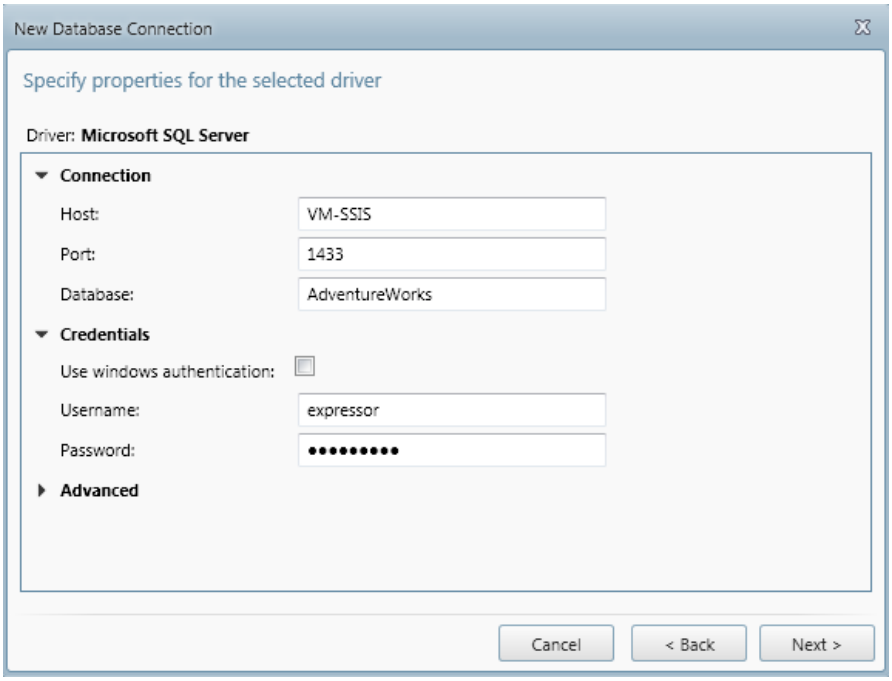
The New Database Connection wizard starts.

1. Select the Supplied database drivers radio button. This is the choice when you are creating a connection for a supported database management system.

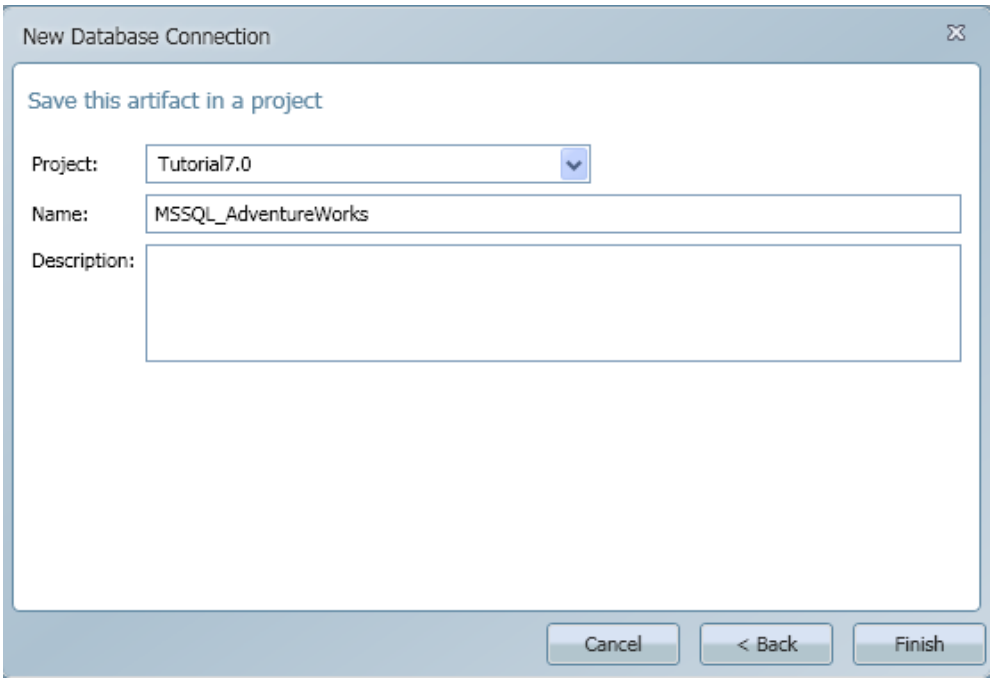
2. From the drop down list, choose the database management system for which you want to create the connection, then click Next.



3. In the following window, enter the Host, Port, Database or database Instance name, and the Credentials of a user authorized to access the database, then click Next. Note that for SQL Server, you may use Windows authentication instead of encrypted credentials. Ignore the controls under the Advanced heading; these are used to modify the connection string in unusual situations.



4. Finally, enter a unique name for the connection and confirm the storage location, then click Finish.



5. The connection appears under the **Connections** sub-folder in your chosen project.

If you want to review the configuration of the database connection, double-click on its icon in the Explorer panel, or highlight the connection, right-click and select Open from the popup menu. Then click Test Connection in the Connection grouping of the Connection > Edit ribbon bar tab to test the connection.

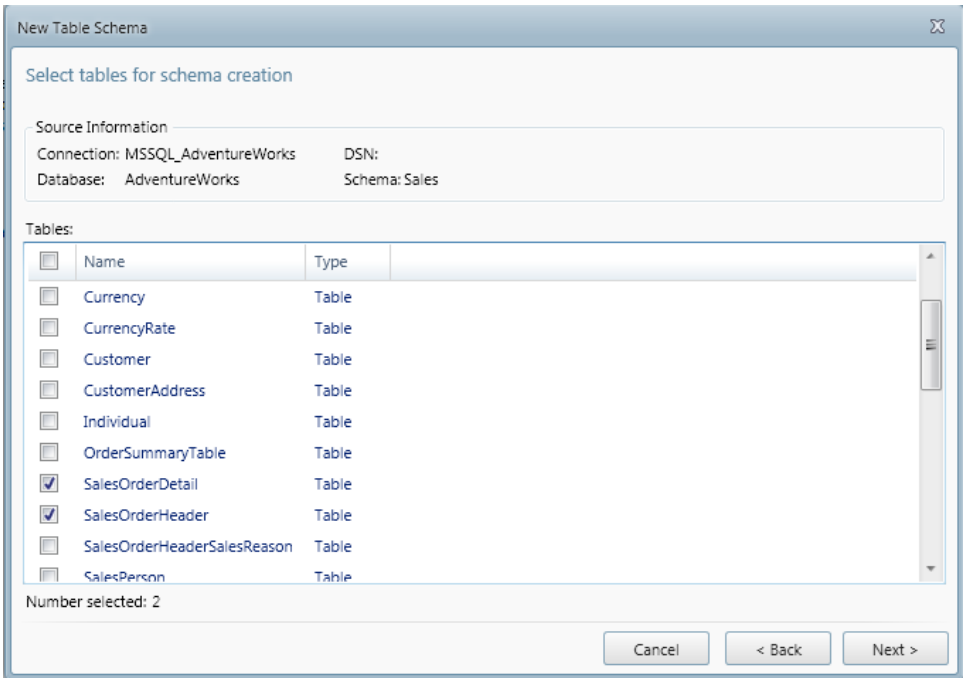
Now that you have the database connection, you can extract the metadata from the database and create a schema for one, or more, tables and/or views. As with the connection, there are multiple ways to start the process to create a schema.

- On the Create tab of the ribbon bar, click Schema > Table Schema.
- In the Explorer panel, right-click on the project and select New Schema > Table Schema > From Data Source from the popup menu.
- In the Explorer panel, expand the project, right-click on the Schemas sub-folder and select New > Table Schema > From Data Source from the popup menu.

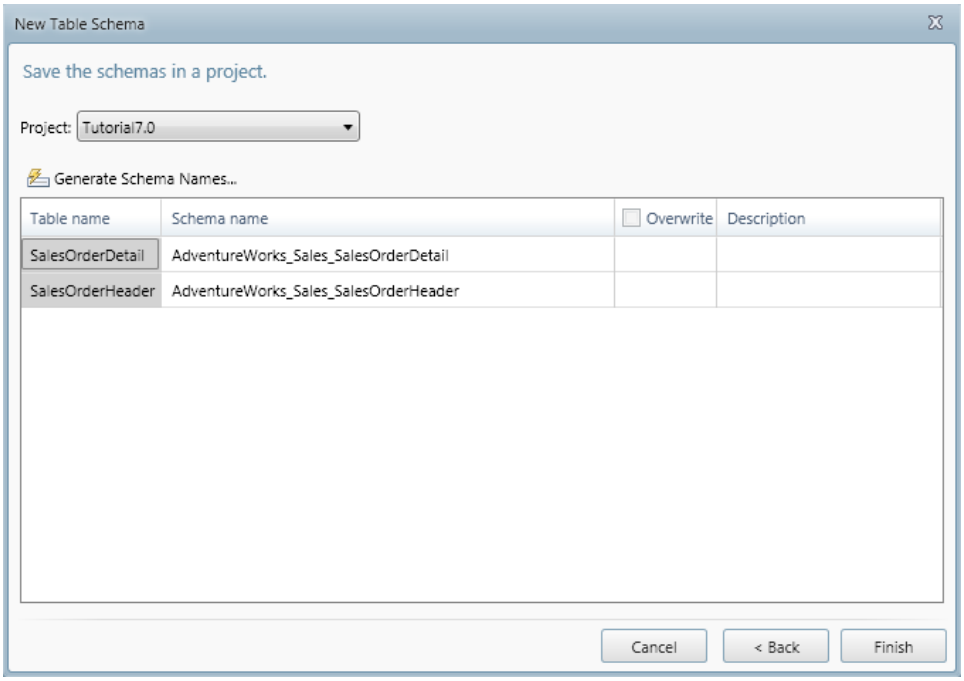
The New Table Schema wizard starts.

1. Note that all existing database connections are already listed. In this case, the only listed connection is the one you just created. If needed, you can create a new connection by clicking the New Database Connection... button in the upper right corner.
2. Select the connection MSSQL_AdventureWorks and click Next.
3. You are now presented with a listing of schemas within the database. Select the desired schema and click Next. For this example, the schema Sales was selected.

4. The next window lists all the tables and views within the selected schema. Select one, or more, entries and click Next.



5. In the last window, confirm the storage location then accept, or change, the suggest schema names. If you are recreating an existing schema, check the Overwrite option so that the existing schema is replaced. Click Finish and the schemas are created in the Schemas sub-folder.



Double-click on a schema's icon in the Explorer panel (or highlight, right-click, and select Open from the popup menu). Note the richer field descriptions in the left-hand column. Both the Data Type and size of each field is recorded.

You may now work within the schemas to create new composite types and mapping sets, promote the

local composite type into a reusable, shareable type, or extend the schema by adding shared composite types and new mappings.

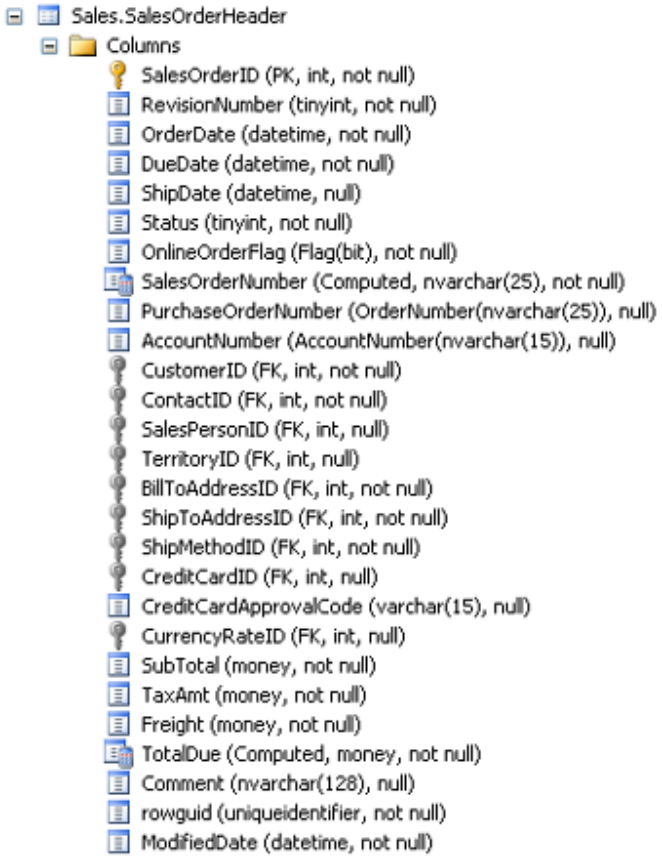
The same database connection and table schemas may be used to read from or write to the tables (or views), and you now have the artifacts needed to configure the Read Table and Write Table operators.

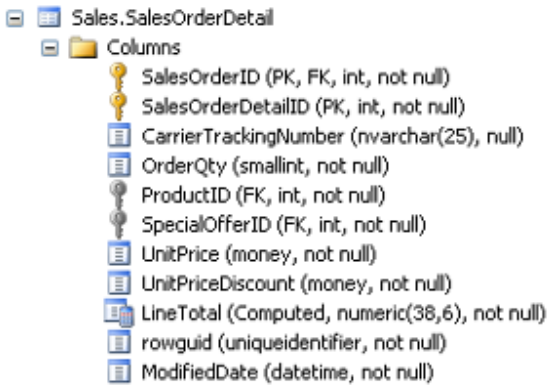
Tutorial 8

Now that you have the artifacts needed to access a database table, what sort of application could you develop that will illustrate database access and usage of some of the other Expressor operators. After all, the objective of this tutorial is to learn about the Expressor not mastery of the AdventureWorks database.

Together the SalesOrderHeader and SalesOrderDetail tables describe a purchase order and its line items. For each record in the SalesOrderHeader table, there are one or more records (line items) in the SalesOrderDetail table. It would be interesting to determine how many line items there are for each order and then divide the orders into groups depending on the number of line items, for example, orders with more than 10 line items and orders with 10 or fewer line items. As output, simply write identifying information for each order to a file.

Briefly review the structure of these tables. The SalesOrderID column is common to both tables; it's the primary key in the SalesOrderHeader table and a foreign key to theSalesOrderHeader table in the SalesOrderDetail table. So this column can be used to join these tables.





For this application, you are not really interested in the table data; you are only interested in how many detail records exist for each header record. This is a perfect situation in which to utilize the power of semantic types. Although the Read Table operators will retrieve all of the columns from each table, you can use the schemas to drop all but the SalesOrderID column — the only information you need to join and count the records. (Want more control over what columns are included in the result set? Read about the SQL Query operator.)

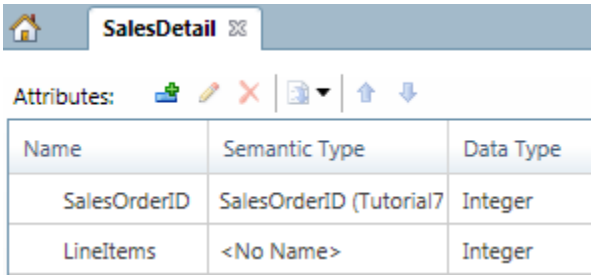
So what additional artifacts are needed?

- A composite type describing the record emitted by each of the Read Table operators. This type needs only one attribute: an integer containing the SalesOrderID. Since the SalesOrderID is central to your application, you will represent this value as a reusable atomic type.
- A composite type describing the record emitted by the Write File operators. This type needs two attributes: an integer containing the SalesOrderID and another integer containing the line item count.
- A schema file describing the structure of the record emitted by the Write File operators. This schema is responsible for mapping the two attribute composite type to the output file.
- A file connection that specifies where to write the output files. This file system location can also be used by any operator that may need to temporarily write data to disk (the join, aggregate, sort, and unique operators have this requirement).

Let's get started!

Step-By-Step Procedure

1. Working within the Tutorial7.0 project, create the file connection specifying the file system location where you want to write the output files (C:\data).
2. Create an atomic type named SalesOrderID, data type Integer. Save this type into the Tutorial7.0 project.
3. Create a composite type named SalesOrderInfo that includes one integer attribute named SalesOrderID. Assign the SalesOrderID atomic type to this attribute. Save this type into the Tutorial7.0 project.
4. Open the schema file for the SalesOrderHeader table in the Schema Editor.
    a. Select the Schema > Edit tab in the ribbon bar.
    b. Click Assign > Add To Schema > Shared in the Semantic Type grouping.
    c. In the Select Shared Composite Type window, select the SalesOrderInfo type and click OK.
    d. Accept the wizard's offer to create a mapping.
    e. Save the modified schema file.
5. Repeat step 4 on the schema file for the SalesOrderDetail table.
6. Create a composite type named SalesDetail that includes an attribute named SalesOrderID and assign the SalesOrderID atomic type to this attribute.
7. Create a second attribute, type integer, named LineItems.
    a. In the Type Editor, change this attribute's type to Integer.
    b. Save your modifications to this attribute and the composite type.

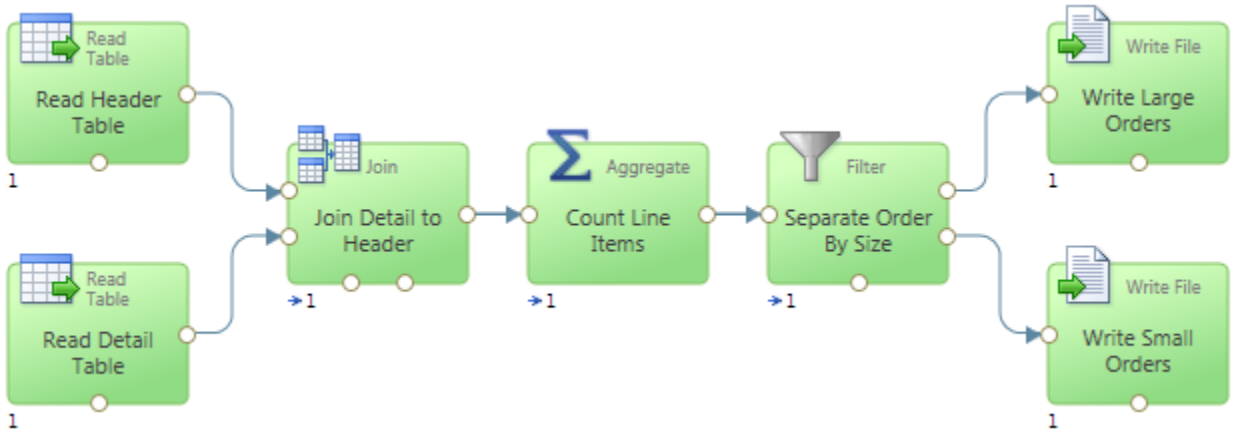


8. Now create a delimited schema from the SalesDetail composite type.
    a. In the Explorer panel, expand the Tutorial7.0 project and the Types sub-folder.
    b. Right-click on the SalesDetail composite type and select the New Delimited Schema From Type… menu entry to launch the wizard.
    c. The type's attributes are displayed. Click Next.

d. In the last step, confirm the location in which to save the schema and give the schema a meaningful name, e.g., SalesDetail. Then click Finish.

Did you realize that step 7 introduced a new concept? You can generate a schema from a composite type.

Now you are ready to design the dataflow, which is named SalesDetail and saved within the Tutorial7.0 project. Your finished dataflow will look like the following figure. Start by placing all of the operators into the dataflow, labeling as shown, and connecting. You can change a shape's label by making an entry into the Name control in its Properties panel.

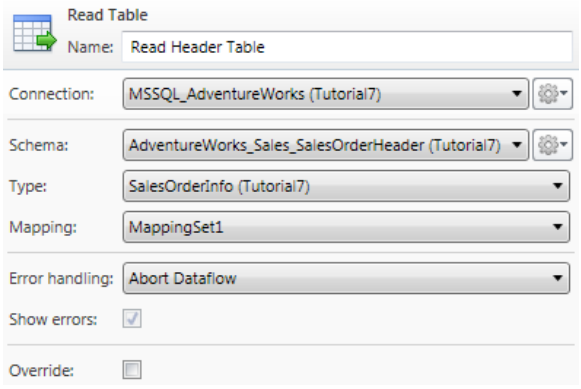


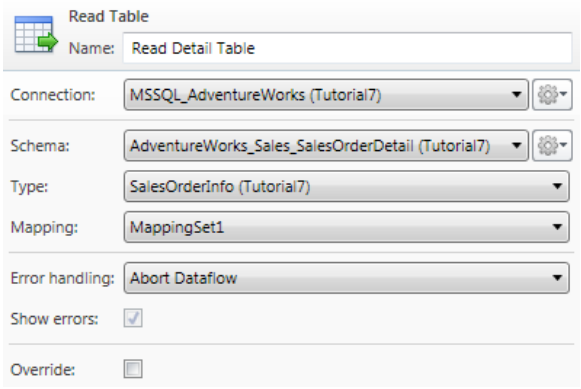
The Join, Aggregate, and Filter operators implement the logic of your application.

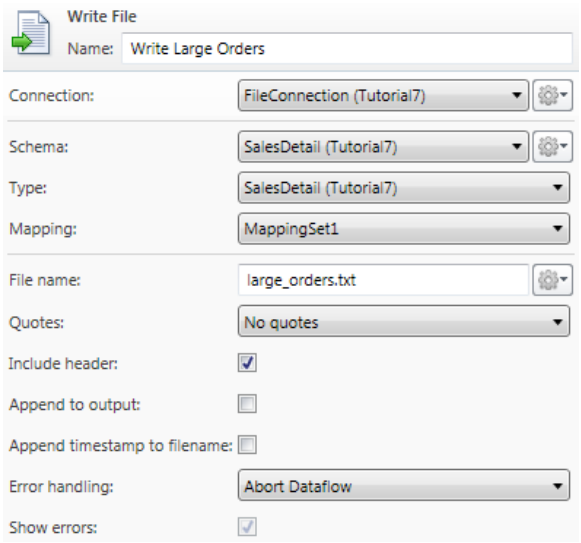
- The Join operator will match a header to its line items, emitting a record each time a match occurs — the classic inner join based on the SalesOrderID. Since a purchase order may have multiple line items, one record will be emitted for each line item in an order. The only data in this record will be the SalesOrderID.
- The Aggregate operator will sum the number of line items for each order. Since we cannot guarantee that the header and line item records are sorted by SalesOrderID, this operator will not emit any records until all of the records have been read by the Read Table operators and received by the Aggregate operator. In this scenario, the Aggregate operator must use RAM or disk space to temporarily store data. If the records were sorted by the join key, no temporary storage space would be required. For each order, this operator will emit one record containing two fields — the SalesOrderID and the number of line items.
- The Filter operator will execute a conditional statement that divides the records into two groups based on the number of line items in the order. Orders with more than 10 line items will be emitted from the upper output while orders with 10 or fewer line items will be emitted from the lower output.

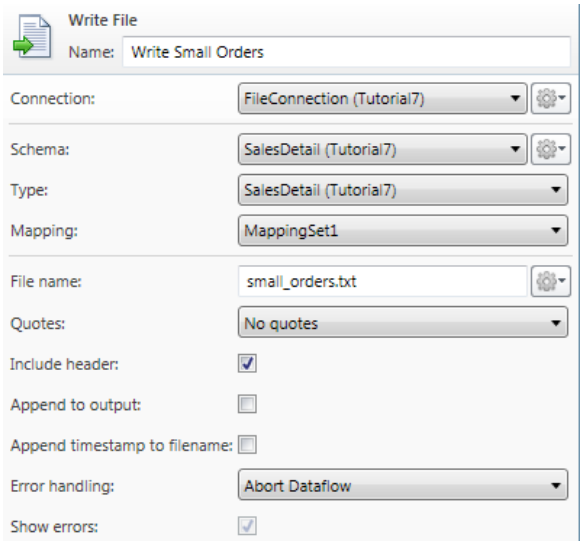
Input and Output Operators

1. The following screen shots show how to configure the Read Table and Write File operators. Note the use of the alternative composite type in the two Read Table operators. Accept the default entries for the controls that are not shown in the following figures.



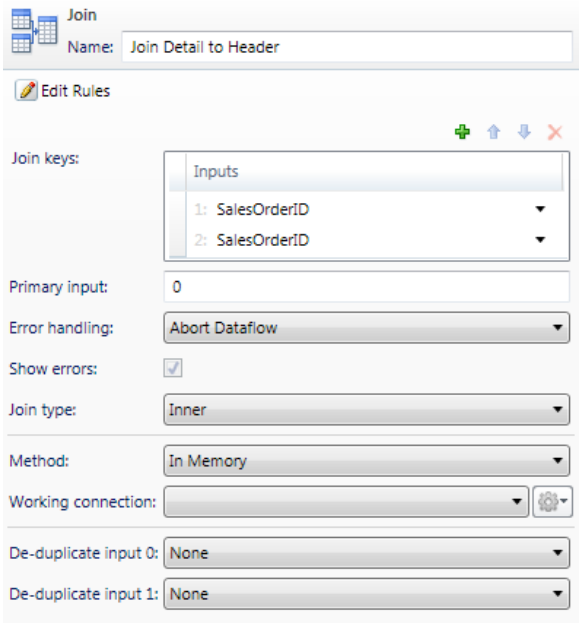






Join Operator

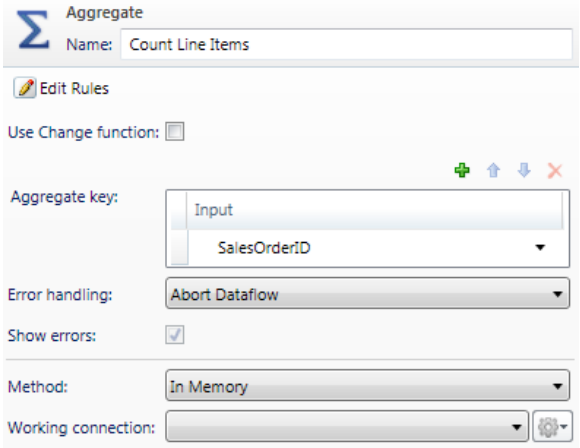
1. Configure the Join operator to perform an Inner join using either RAM (In Memory) or disk (On Disk) for temporary storage. The records retrieved from the SalesOrderHeader table (Primary input: 0) should drive the processing. Note: Accept the default entries for the controls that are not shown in the following figures.



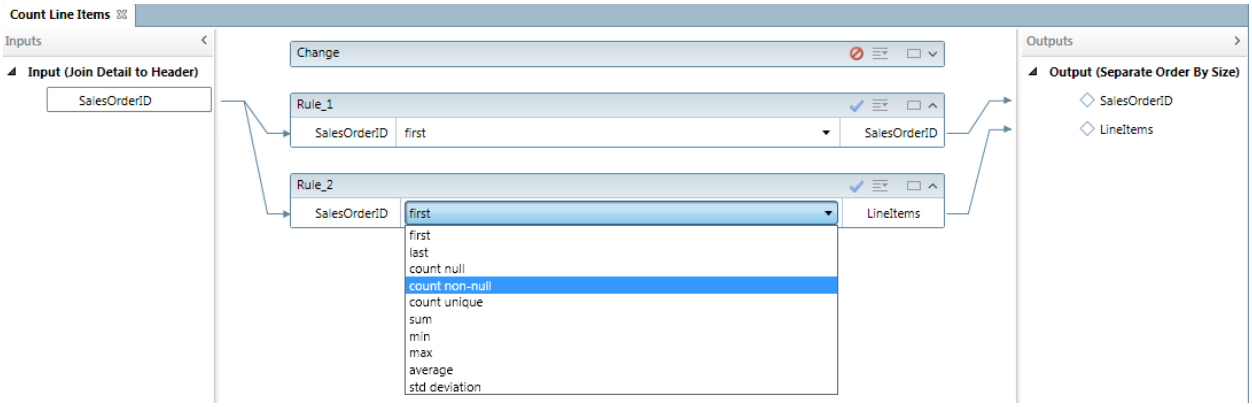
2. Click on the Edit Rules link.
3. Note that each incoming record has only one field, SalesOrderID. This is exactly what you wanted. By specifying the alternative composite type in the schemas assigned to the Read Table operators, you instructed the operators to drop all other fields and only include the SalesOrderID value in the emitted record. This functionality greatly eases your subsequent effort.
4. Also note that a single attribute appears in the output panel. You can tell by the right-facing arrow that this attribute will be transparently initialized by Expressor's attribute propagation, which is exactly what you intend, so there is nothing else you need to do.
5. Close the Rules Editor. Your work in this editor is automatically saved.

Aggregate Operator

1. Configure this operator to use SalesOrderID as the key. This means that the incoming records will be collected into groups sharing the same SalesOrderID value. Again, use either RAM (In Memory) or disk (On Disk) for temporary storage.



2. Click on the Edit Rules link. Observe that the output panel includes two attributes and you can tell from the left-facing arrows that these attributes have been pushed up into this operator from the downstream operator. You will need to create the rules that initialize these attributes.
3. Using your mouse, drag from the SalesOrderID input attribute to the SalesOrderID output attribute. An Aggregator Rule is automatically created and the aggregate function first selected. This is the aggregator function you need for this attribute, so there is nothing more to do.
4. Using your mouse, drag from the SalesOrderID input attribute to the LineItems output attribute. Again, an Aggregator Rule is automatically created, but for this rule you want to choose the count non-null aggregator function.



5. Close the Rules Editor. Your work in this editor is automatically saved.

Filter Operator

1. Open this operator's Rules Editor. You must provide code that returns either true or false. Return true if the order has more than 10 line items and false if the order has 10 or fewer line items.
2. When you first open the editor, the operator is coded to always return true.

3. Drag the attribute LineItems from the input panel into the rule, then enter the statement input.LineItems > 10.

4. Close the Rules Editor. Your work in this editor is automatically saved.

Run the Dataflow

1. Save the dataflow.
2. To run, click Start in the Run grouping in the Dataflow > Design tab of the ribbon bar.
3. There will be two output files.
    a. The file small_orders.txt will contain a list of order numbers where the number of line items is 10 or fewer.
    b. The file large_orders.txt will contain a list of order numbers where the number of line items is greater than 10.

Tutorial 9

In the last two tutorials, you followed a tightly scripted development plan in designing the data integration application. You were able to create all of the required schema and type artifacts before you began creating the dataflow.

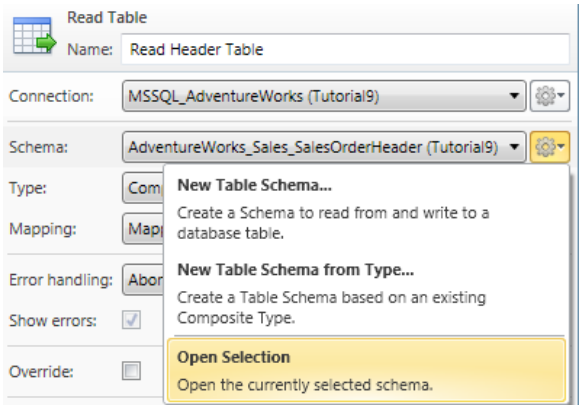
With many applications, you may not be able to plan as carefully as in these exercises before you begin work on the dataflow and there are instances in which you want to immediately begin laying down the dataflow, defining connection, schema, and type artifacts as you proceed. The left-to-right development paradigm addresses these alternative approaches.

With the approach illustrated in the earlier tutorials, you actually developed the application from the outside in; that is, you first created artifacts, then configured the input and output operators, and finally worked on the intervening operators. With the left-to-right approach, you create each artifact when it is needed while developing your application logic from start to finish. The terminology left-to-right comes from the fact that in Expressor dataflows processing runs from each operator to the operator on its right. In this scenario, artifacts are created for each operator before moving to its downstream, right-side, operator.

As an example of this approach, let's redevelop the application of tutorials 7 and 8.

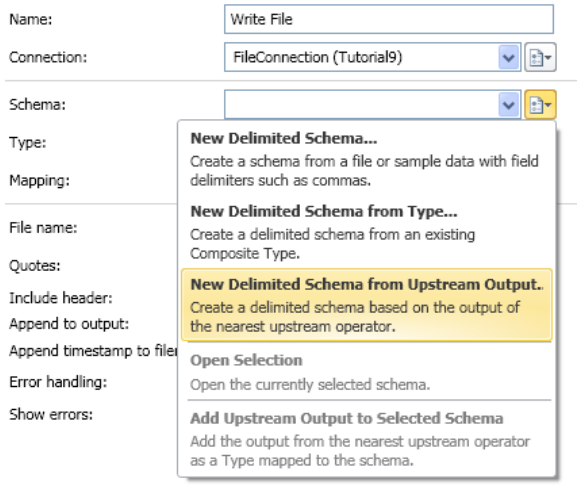
Step-By-Step Procedure

1. Create a new project named Tutorial9.
2. Within this project, create a new dataflow named Tutorial9.
3. Add a Read Table operator to the dataflow. From its Properties panel, create the required connection and schema artifacts.
    a. Create the schema for the Sales.SalesOrderHeader table.
    b. Open the schema and create a second composite type with only a single attribute — SalesOrderID. A shortcut way to open the Schema Editor is to select Open Selection from the drop down menu accessed from the button adjacent to the Schema control.



    c. Use this second composite type when configuring this Read Header Table operator.
4. Add a second Read Table operator to the dataflow. From its Properties panel, reuse the connection and create a schema.
    a. Create the schema for the Sales.SalesOrderDetail table.
    b. Open the schema and create a second compostie type with only a single attribute — SalesOrderID.
    c. Use this second composite type when configuring this Read Detail Table operator.

5. Add and connect the Join operator. Configure an in memory inner join using the SaleOrderID attribute as the join key.
    a. Create a file connection to be used as the working connection.
    b. Open the Rules Editor and note that the attribute in the output panel will be initialized through Expressor's attribute propagation functionality. You do not need to do any other coding.
    c. Close the Rules Editor.
6. Add and connect the Aggregate operator. Configure for in memory operation using SalesOrderID as the key.
    a. Open the Rules Editor and note that there are no attributes defined for the output.
    b. In the Output Attributes grouping, click Add and create an output attribute named SalesOrderID, type integer.
    c. Then create a second output attribute named LineItems, type integer.
    d. Using the mouse, drag a line from the input attribute SalesOrderID to the output attribute SalesOrderID. An Aggregator Rule using the aggregate function first is automatically created.
    e. Again using the mouse, drag a line from the input attribute SalesOrderID to the output attribute LineItems. In the Aggregator Rule, select the aggregator function count non-null.
    f. Close the Rules Editor. Your work is automatically saved.
7. Add and connect the Filter operator.
    a. Open the Rules Editor and add the required coding to the filter operator.
    b. Drag the input attribute LineItems into the rule then enter the coding statement `input.LineItems > 10`.
    c. Close the Rules Editor, Your work is automatically saved.
8. Add and connect two Write File operators.
    a. Select one of these operators and select the file connection.
    b. To create a delimited schema, select New Delimited Schema from Upstream Type from the drop down menu accessed from the button adjacent to the Schema control.



    c. In the first New Delimited Schema from Type window, select the upstream type and click Next.
    d. In the final window, name the schema OrderDetailSchema and click Finish.
9. Once the schema is complete, enter a name for the output file into the File name control.
10. Select the other Write File operator. Make selections for the Connection, Schema, Type, and Mapping controls. Enter a name for the output file into the File name control.
11. Save and run the dataflow.

Tutorial 10

This tutorial replicates the data integration application developed in tutorials 7, 8 and 9 using file, rather than database, input. This document will only describe the setup and configuration of the Read File operators. Once you have completed this tutorial, return to either tutorial 8 or 9 to complete the data integration application.

Step-By-Step Procedure

1. In workspace_2, create a new project named Tutorial10.
2. Within this project, create a new dataflow named Tutorial10.
3. Add a Read File operator to the dataflow. From its Properties panel, create the required file connection and schema artifacts.
    a. Enter Read Header File as the operator's name.
    b. Create the schema for the file salesOrderHeaders.txt, which is in the directory C:\data.
    c. Open the schema and create a second composite type with only a single attribute — SalesOrderID. Change the type of this attribute to Integer.
    d. Use this second composite type when configuring the Read Header File operator.
4. Add a second Read File operator to the dataflow. From its Properties panel, reuse the file connection and create a schema.
    a. Enter Read Detail File as the operator's name.
    b. Create a schema for the file salesOrderDetail.txt, also in the directory C:\data.
    c. Open the schema and create a second composite type with only a single attribute — SalesOrderID. Change the type of this attribute to Integer.
    d. Use this second composite type when configuring the Read Header File operator.
5. Return to Tutorial9 and begin working at **Step 5**.