# Table of Contents

**What Is Data Process Logic Studio?**  
Quick start: A sample Data Process Logic Studio project  
Quick start 1: Identify input sources  
    To identify your data sources  
Quick start 2: Define selection criteria  
    To define data selection criteria  
Quick start 3: Transforming the Data  
Quick start 4: Export templates  
    To create and run an export template
Understanding Data Process Logic Studio

Understanding the Data Process Logic interface .......... 28

Document area.......................................................... 30
Data pane........................................................................ 30
Dynamic Help pane....................................................... 31
Toolbar........................................................................... 31
Tools menu........................................................................ 31

Understanding Data Process Logic packages .............. 38
Sample package.............................................................. 38

Understanding the data flow graph ......................... 39

Installing Data Process Logic Studio 47

Understanding system requirements.......................... 48
Installing Data Process Logic Studio.......................... 49
To install Data Process Logic Studio......................... 49

Understanding the Disk Cost tool................................. 56
 Licensing Data Process Logic Studio......................... 58
To license Data Process Logic Studio......................... 58
To upgrade an evaluation license to a permanent license ................................................................. 59
Data sources: Connections

Creating data connections.................................61
  To create a data connection (general steps)........61
Creating Data Repository ........................................67
  To add a data connection to the data repository ......68
Creating Virtual Connection ....................................74
  To create a virtual connection: .......................74
Associate Data Models ........................................79
  To associate data models: ............................79

Data Selection: Queries

Creating Standard Queries ...................................86
Creating Standard Queries in Design Mode ...............87
  To create a query in design mode ......................88
Creating Queries in Advanced Mode ......................98
  Create a query in advanced mode .....................99
Using the Expression Editor .................................100
  To use the Expression Editor .......................100
Creating Data Model ........................................104
Understanding aggregate queries ......................107
Creating aggregate queries ........................................ 108
   To create an aggregate query .................................. 108
Understanding transform queries ............................... 112
   Sample transform queries ..................................... 113
Creating transform queries ...................................... 117
   To create a transform query ................................... 117
**Validation Rules** 119
   Steps to create Validation Rule ................................. 121
**Data Export: Templates** 125
   To create an export template .................................. 126
   Setting up Access destination connections ............... 132
     To set up a Microsoft Access destination connection 132
   Setting up Excel destination connections ................. 133
     To set up a Microsoft Excel destination connection ... 133
   Setting up MySQL destination connections ............... 134
     To set up a MySQL destination connection .............. 134
   Setting up OLE DB destination connections ............... 136
     To set up an OLE DB destination connection .......... 136
   Setting up Oracle destination connections ............... 138
     To set up an Oracle destination connection .......... 138
Setting up SQL Server destination connections ............ 139
  To set up an SQL data connection ............................... 139
Setting up Text destination connections ......................... 141
  To set up a text connection .................................... 141
Setting up XML destination connections .......................... 143
  To set up an XML data connection ............................... 143
Setting up SAP Business One- Business Object destination connection.................................................. 144
  To set up an SAP Business One-Business Object data connector .......................................................... 144
Setting up SAP XML IDoc destination connection ............. 146
  To set up an SAP XML IDoc connector ........................... 146
Setting up SAP SII destination connection ....................... 147
  To set up an SAP SII data connector ............................ 147
Setting up SAP destination connection ............................ 148
  To set up an SAP data connector .................................. 148
Setting up WebDB Server destination connection ............. 149
  To set up an WebDB Data connector ............................. 149
Exporting data: running an export template .................... 151
  To export data ...................................................... 151
Creating Quick Export .................................................. 152
To create a quick export ........................................... 152
Using Command-Line Export ........................................ 156
Integration Functions Overview ...................................... 163
Integration Function API Specification ........................... 164
Using the Integration Function Editor ............................. 172
Integration Function Examples ....................................... 175
User Functions Overview ............................................. 179
User Function API Specification .................................... 180
Using the User Function Editor ..................................... 182
User Function Examples .............................................. 183

Using Data Process Logic as a data bridge .......................... 187

Configuring Data Process Logic as an OLE DB data source
................................................................................... 189
Example: Using Data Process Logic as a data bridge with
Microsoft® Excel ......................................................... 190

Data Process Logic Studio as a data bridge To use to
Excel ............................................................................... 190
Appendix A Expression syntax

Expression syntax: data types .................................................. 194
Type conversion rules .............................................................. 194
Expression syntax: grammar components .................................. 196
  Literal s ............................................................................ 196
  Table s .............................................................................. 196
  Column s ........................................................................... 196
Expression syntax: operators ..................................................... 198
  Arithmetic operators ............................................................ 198
  Relational operators ............................................................. 199
  Logical operators ................................................................. 199
Expression syntax: functions ................................................... 201
Expression syntax: other language features ................................. 203
  Arrays .............................................................................. 203
  If-Then-Else ..................................................................... 203
  Data Process LogicTestTable ................................................. 204

Copyright and Trademarks ....................................................... 205
What Is Data Process Logic Studio?

Data Process Logic Studio is a software platform for integrating data from various data sources. With its powerful yet easy to use functionality, Data Process Logic Core Engine helps take the complexity out of data integration. The platform can extract and aggregate data across multiple sources, apply powerful data transformations and migrates high volume of data to virtually any other data handling application. Data Process Logic Core Engine can even be used as a back-end to existing reporting systems such as Microsoft® Reporting Services.

Data Process Logic Core Engine is utilized by three programs:

Data Process Logic Studio is a front end Windows application for Data Process Logic Core Engine.
Data Process Logic Export is a command line application that can be used along with any standard scheduler to automate data processing tasks.

Data Process Logic WebDB Server is a web-based application that facilitates secure communication over the Internet among business partners and remote offices.

Data Process Logic Studio easily integrates, analyzes, and exports data, turning raw data into business intelligence. The process involves only four main tasks.

1. **Connection**: Identify input sources by creating data connections (detailed in the chapter Data sources: Connections).
2. **Extraction**: Specify data selection criteria with queries (covered in the chapter Data selection: Queries).
3. **Transformation**: Convert data to a format that is suitable for the destination data source. During this step, you can also cleanse your data.
4. **Loading**: Create and run export templates to send data to other sources (covered in the chapter Data export: Templates).
Quick start: A sample Data Process Logic Studio project

The rest of this chapter works through the four main steps while explaining the interface and components within Data Process Logic Studio. The information is presented within the context of a real-world example combining sales data from three regional offices into a single table where all revenue figures are normalized to US dollars.

This project uses sample data files that are installed with Data Process Logic Studio. All data files are found in the SampleData subdirectory under the Data Process Logic Studio installation directory. If you prefer, a video demonstration of this example is on the Data Process Logic CD-ROM or you can download it from the customer portal.

Before working through the following chapters, be sure you have installed Data Process Logic Studio using the steps described in the chapter titled Installing Data Process Logic Studio.
Quick start 1: Identify input sources

The first steps in working with Data Process Logic Studio are to create an Data Process Logic package (that is, a file to hold all the components of your project) and to identify your input sources by creating data connections. For more information about creating connections, see Data sources: Connection

To identify your data sources

1. Create an Data Process Logic package file by going to the File menu and selecting New. The interface changes to show, that you now have a file ready to receive new connections, queries, and export templates. (For details about the Data Process Logic Studio interface, see Understanding the Data Process Logic interface)

2. Identify the file containing product information by creating a data connection.
   - In the Data pane, right-click Connections and select New Connection from the shortcut menu.
   - You can also click on Action: Create a connection to your data on the Dynamic Help pane.
   - In the Select a connection type dialog, click to choose XML Data Connector.
   - In the Select XML files dialog, click on the Add Files and navigate to Program Files\RISE\4.3\DataPro Logic Studio\SampleData\Products.xml, then click Open.

---

Data Process Logic Studio User's Guide
- **Data Process Logic Studio** can create a schema for an XML file, or you can import a pre-existing schema in an XSD file. For this example, click **Import Schema**, select **Products.xsd**, and click **Open**, and then click **Finish**.

- Back in the main interface, go to the **Data** pane, right-click on the **New Connection** you have just created, and select **Rename** from the shortcut menu. Type “Products” as the new name.

3. Use the same procedures to identify the files containing sales information.

- Create a **New Connection** for the file **EuropeSales.xml** using the schema in the file **Sales.xsd**. **Rename** the connection “Europe Sales”.

- Right-click the “Europe Sales” connection and choose **Duplicate** from the shortcut menu. Right-click the “Copy of Europe Sales” that is created and select **Edit** from the shortcut menu. In the **Select XML files** dialog, change the file by using **Remove** on the Europe Sales file, then use Files to select “JapanSales.xml”. Rename the connection “Japan Sales”.

- Either create a **New Connection** or **Duplicate** one of the existing connections to create a “USA Sales” connection using the “USASales.xml” and “sales.xsd”.

---

**Data Process Logic Studio** Troubleshooting Guide 13
4. Finally, create a connection to identify the file with the currency conversion rates that will be used to normalize the sales data.

- In the Data pane, right-click Connections and select New Connection from the shortcut menu.
- In the Select a connection type dialog, click on ISAM/Files tab to choose Text Data Connector.
- In the Open Text file dialog, click the Files… link and navigate to Program C:\Program Files\RISE\4.3\Fusion Studio\SampleData\CurrencyRates.txt. Click Open then Click Next.
- In the Open Text file dialog, click on AutoDetect to detect the file by Data Process Logic Engine and place the file format on the Name filed.
- In the Code Page text box field, you can specify the code page.
- In the Open Text file dialog, click on Load Column Definitions to load column definition (if required), then click Next.
- When using text files as data sources, you must specify how to interpret the data. For this example, click Delimiter for the original data type then click Next.
- Select Tab as the Delimiter type, and then click Next. In the Select column data type dialog, select the Currency column and select String as the data type. Select the Rate column and select Double as the data type.
- Click **Finish**.
- Back in the main interface, go to the **Data** pane, right-click **New Connection** you have just created, and select **Rename** from the shortcut menu. Type “Rates” as the new name.

5. Drag the connections on the screen so that you can see all of them. The screen should look something like the image below.

6. Save your entire **Data Process Logic** package by going to the **File** menu, selecting **Save**, and type “Sample” as the file name. Click **OK** to close the dialog.
Quick start 2: Define selection criteria

Now that you have identified the files that will hold your input data, you will use Data Process Logic Studio queries to specify which information in the files will be used and how it will be combined or transformed. Because the source files in this example are in formats (XML and text) that do not have a native query language, our queries will use Data Process Logic Studio design mode queries. For more information about creating queries, see Data selection: Queries, p 69

To define data selection criteria

1. Begin creating a query that will gather all the information about the available products.
   - Open a new query by going to the Data pane, right-click on Queries and select New Standard Query (OR) right click on Document Area and select New standard query.
From the toolbar at the top of the screen, use the drop-down list to select **Products** connection as the basis of your new query.

Drag the **Products** table from the upper right quadrant (**Tables** section) to the **Content** pane (center section, just below the **New Standard Query** tab).

Click on the asterisk (*) in the lower left corner of the **Products** table to select all fields in the file.
Click the **Preview** icon on the toolbar or press **F5** from keyboard to view the information that your query gathers.

- In the **Data** pane (lower left), right-click on **New Standard Query**, select **Rename** from the shortcut menu, and type “Products” as the new name.
- Save the query by going to the **File** menu and select **Save Products**.

2. To gather the sales data, repeat the tasks in step 1 to create each of the following additional standard queries. Use the **Preview** icon and note that each query returns sales figure in the native currency (e.g. “Japan Sales” returns figures in yen).
   - “USA Sales” using the “USA Sales” connection and all fields from its “Sales” table.
   - “Japan Sales” using the “Japan Sales” connection and all fields from its “Sales” table.
   - “Europe Sales” using the “Europe Sales” connection and all fields from its “Sales” table.

3. To gather the currency rates data, create a standard query that prompts you to select the correct currency for conversion when the query is run.
   - Go to the **Data** pane, right-click on **Queries**, and select **New Standard Query** from the shortcut menu.
   - Use the drop-down list on the toolbar to select the **Rates** connection as the basis of your new query.
- Drag the **Table** from **Tables** pane to the **Content** pane.

- Click the **asterisk** (*) in the lower left corner of the Table to select all fields in the file.

- To add the prompt for the parameter, right-click on the **Currency** field in the **Table** and select **Add Filter** from the shortcut menu.

- In the text box following **Table.Currency**, type ‘{Currency}’. The quotation marks signal that this filter uses a string and the braces signify that the query should prompt for the parameter when the query is run. Close the **Query Properties** dialog by clicking the **X** (Close) in the upper right corner.

- Verify that the **SQL** pane reads as follows:

  ```sql
  SELECT Table.Currency, Table.Rate
  FROM Table WHERE Table.Currency = '{Currency}'
  ```

- In the **Data** pane (lower left), right-click on **New Standard Query**, select **Rename** from the shortcut menu, and type “Rates” as the new name.

- Save the query by going to the **File** menu and selecting **Save Rates**.

4. Next we’ll create a transform query that transforms the Japan sales data in to U.S. dollars. The transform queries of **Data Process Logic Studio** can perform operations on data from other
queries. For more information, see Understanding transform queries.

- Go to the Data pane, right-click on the Queries, and select New Transform Query from the shortcut menu.
- Transform queries can be typed directly into the Content pane or you can drag items from the Queries and Functions pane, list at the right. First, type “Select ()” into the Content pane, then drag each of the fields under the “Japan Sales” query between the parenthesis and adding a comma after each field. The Content pane should look like this:
  
  ```
  Select ([Japan Sales.Country], [Japan Sales.ModelNumber], [Japan Sales.Margin], [Japan Sales.Price], [Japan Sales.UnitsSold])
  ```

- Click the Preview icon or F5 to verify this query returns data from the file.
- The transformation we want to add to this query is to divide the price by the exchange rate. Click in the Content pane after the square bracket enclosing the [Japan Sales.Price] field and type the following expression to divide the price by the rate for yen:

  ```
  o SelectValue ([Rates ("Yen").Rate], 1)
  ```

- Because you have modified the Price column, Data Process Logic will treat it as a new column. We want to retain the column name, so you must use the “as” keyword to explicitly assign the name “Price”.

---

Data Process Logic Studio User’s Guide
The entire query should look like this:

```
Select ([Japan Sales.Country], [Japan Sales.ModelNumber], [Japan Sales.Margin],
        ([Japan Sales.Price]/SelectValue([Rates("Yen").Rate],1)) as Price, [Japan Sales.UnitsSold])
```

- Click the **Preview** icon or F5 to verify the query returns the price in US dollars.
- Rename the query “Normalized Japan Sales” and save it.

5. Repeat step 4 to normalize the European sales in the same way. Remember to use “Euro” for the rate parameter instead of “Yen”.

6. Next, we'll combine all sales data into a single table by creating a transform query using the AppendTable function.

- Go to the **Data** pane, right-click on **Queries**, and select **New Transform Query** from the shortcut menu.
- In the **Content** pane, type “AppendTable (“ (without the quotes), then drag the Normalized Japan Sales, Normalized Europe Sales, and USA Sales queries to the pane, adding a comma after the first two query
names and “)” (without the quotes) at the end. Your query should look like this:

```
AppendTable ([[Normalized Japan Sales]],
[[Normalized Europe Sales]], [[USA Sales]])
```

- Click the **Preview** icon or F5 to verify the query returns all the data.
- Rename the query **Unified Sales** and save it.

7. Note that the sales data includes only the model number, but no additional information about the product. We’ll create a final query, an aggregate query, to add more details from the Products query.

- Go to the **Data** pane, right-click on **Queries**, and select **New Aggregate Query** from the shortcut menu.
- Drag the **Products** and **Unified Sales** queries to the **Content** pane, and click the asterisk (*) in the lower left corner of each query to select all fields.
- To create a join between the two queries, drag from the “Model Number” field in the **Product** query to the field with the same name in the **Unified Sales** query.
- Click the **Preview** icon to verify the query works. You may use **Stop fetching data** icon on the toolbar to stop fetching data if the query returns a large amount of data.
- Rename the query **Unified Product Sales** and save it.
Your query should look like the following graphic.
Quick start 3: Transforming the Data

Transform queries use Data Process Logic Studio expressions to manipulate and transform existing queries (standard, aggregate and transform) to produce new data and new data formats. Expressions are a way to use functions and logic with data. Transform queries are created using the transform query editor.

The final value of a transform query is always presented as a table, regardless of the dimension of the source expression.

1. **If you create a transform which uses or returns a scalar (individual) value**, the transform produces a table with a single column and single row containing that value.

2. **If you create a transform which uses or returns a vector (one-dimensional series) value**, the transform produces a table with a single column displaying that vector.

3. **If you use a transform within another transform**, it will always be treated as a table and will return a table as the result.
Quick start 4: Export templates

Now that you have selected, transformed, and aggregated your data, you will use Data Process Logic Studio export template to export the data to a new file. For more information about creating queries, see Data Export: Templates.

To create and run an export template

1. Create a new export template by going to the Data pane, right-clicking Export Templates and select New Template from the shortcut menu.

2. In the Set destination connection dialog, click on ISAM/Files and select to choose XML Data Connector.
   - In the Select output file name dialog, type Archive.xml in the Filename field to have Data Process Logic Studio creates a new XML file named Archive.xml or click on File… to browse the existing XML file, then Click Next.
   - In the Define table mappings dialog, click Add new mapping to define the correlation between the Data Process Logic query data and the export file.
   - In the Select data source dialog, click to select the Unified Product Sales query, that we’ve just created.
   - In the Select update option dialog, type the log file name in the Log file field and click Next.
- In **Validation and Notification** dialog, select the dropdown list to display all the validation rules defined in the fusion pack. By default no rule is selected.

- If we define any validation rule in the fusion pack, in Output file box, it will display the filename as `%FPDIR%\Validation.htm` and it will create that file in a same directory where we did saved the fusion pack only if the validation rule got failed.

- If you want to create a output file in different directory, by clicking on (...) beside to the Output file box.

- Specify your email id, if you want the notification about export fails or succeeds. Then click **Next**.

- In the **Summary** dialog, click **Run Export** to run the export.

- Click **Finish** to exit the wizard, and then rename the **Export Template**.
Understanding Data Process Logic Studio

This chapter explains the core concepts and components of Data Process Logic Studio. The information in this chapter is intended to provide a high-level description of the entire Data Process Logic Studio application and a basic description of common tasks. For specific information about connections, queries, export templates, and Data Process Logic Studio functions, see the individual chapters covering each topic.
Understanding the Data Process Logic interface

Data Process Logic Core Engine runs within a framework called Data Process Logic Studio. Data Process Logic Studio provides a single integrated environment to run Data Process Logic and any other Data Process Logic modules. As described below, three main areas appear in the workspace by default: the Document area, the Data pane and the Dynamic Help pane. A toolbar spans the top of the application, and additional specialized functionality is available from the Tools menu.

When you first open Data Process Logic Studio, the Data pane is empty. The Connections, Queries, and Export Templates folders shown below appear when you create or open a Data Process Logic package (*.slfusion)
Document area

The main section is the **Document area**, which can contain multiple documents. Each document is accessible by clicking on the tabs at the top of the pane (The graphic shows a document area with a single **untitled** document open.) To close a document click the X at the top right of the pane.

Note: A tabbed document does not necessarily represent a single *.slfusion file. **Data Process Logic Studio** uses child documents to edit or view various aspects of the package. Only the parent document actually represents a file. When saved, tabbed child documents commit their changes to the parent document. **Data Process Logic Studio** saves the physical file only when you save the parent document, which also forces any open child documents to save their changes first.

Data pane

The **Data** pane on the bottom left of the application contains all available connections, queries, and export templates. Each item appears as a folder. Within this pane, you can navigate to most of the package elements and perform basic functions:

- **To create a new item** (connection, query, or export template), right-click the appropriate folder and select the item you wish to create.

- **To rename an item**, select the item and press **F2** or right-click on the item and select **Rename** from the shortcut menu.
To delete an item, select the item and press **Delete** or right-click on the item and select **Delete** from the shortcut menu.

To open the editor for an item, double-click on the item.

**Dynamic Help pane**

The **Dynamic Help** pane provides context-sensitive help for the current document and tasks. Clicking on the hyperlinks in the pane either directs you to a topic within the on-line help, or performs an action for you.

To hide this pane click the ‘X’ on the top right of the **Dynamic Help** pane. To reinstate the pane, select **Dynamic Help** from the **Help** menu or press **Alt H+D** from the keyboard.

**Toolbar**

The toolbar contains tools to perform common actions within the current module. Tools that are not available in the current context are disabled and appear gray. Dragging the toolbar turns it into a floating tool window that can be placed anywhere on the screen.

**Tools menu**

The **Tools** menu provides access to four **Data Process Logic Studio** specific functions:
Status
Displays the status console, containing time stamps and status messages from Data Process Logic Studio. This is useful for troubleshooting complex Data Process Logic packages which require specific details about how the process is executing.

Configuration:

1. Data Fetch-
   It supports for Ms Access 2003 or Ms Access 2007.
   Provides settings to adjust performance:
   - Max rows fetched on preview: Default position for the max row slider in the preview pane.
   - When using Access, the number of columns that can be selected are limited to 255 columns
   - Max time duration on preview: The number of seconds Data Process Logic attempts to fetch preview data before timing out.
   - Temporary data store path: Folder where Data Process Logic Studio stores its temporary data cache.

2. SQL Server:
   It supports only for SQL Server 2005.
   - Check the checkbox Use SQL Server,
- Give **server name, username and password** for test connection.

**Flush local data cache**

*Data Process Logic Studio* caches all data locally to improve aggregation performance and to reduce the load on the data providers. Flushing the local data cache forces *Data Process Logic Studio* to query the data provider once again. This is useful when a change has been made in the underlying data source or the data within *Data Process Logic Studio* is out of date.

**Data Storage:**

To reduce round-trips to the data source, *Data Process Logic Studio* caches the data that has been fetched from the input data source(s). This data is stored in an area called data storage. The data storage is essentially a database for caching data as well for performing joins across heterogeneous data sources.

More detailed differences between Microsoft Access and SQL Server 2005 from Data Process Logic Studio perspective can be found within the knowledge base at the customer portal site.

There are three types of cached tables – temporary, semi-permanent and permanent. Temporary tables are used by text and XML connectors as well as some transform functions. These tables are created temporarily and are deleted after its use.

When complete data for a query is fetched, Data Process Logic Studio keeps this data cached. This cached data is referred to as semi-permanent tables. Semi-permanent tables remain cached until one of the following conditions happen:

1. You quit about of Data Process Logic Studio. In this case, all the semi-permanent queries are automatically deleted.
2. You explicitly flush the semi-permanent table.
3. You invoke “flush all” command. This causes the data storage to be completely emptied.

If the fetched data does not change that often, you can mark a cached table as “permanent.” A permanent table remains cached until one of the following conditions happen:

1. If you are using Microsoft Access for data storage and you quit out of Data Process Logic Studio.
2. If you explicitly flush the permanent table.
Note that if you are using SQL Server 2005 as the data storage, a permanent table remains cached even after you quit Data Process Logic Studio.

Also note that, when you invoke “flush all” command, permanent tables are not flushed. Only explicit flush can remove permanent tables.

Each Data Process Logic package has one data store associated with it. To improve efficiency, the data store is created only if your action causes data to be fetched (such as, running a query).

To ensure that cached values are not used during data exports, any of the following operation will result in creating a new data store:
- Running a quick export
- Running an export template
- Running an integration function
- Running DB Copy

This newly created data store automatically gets deleted at the end of the operation.
Some features for data storage UI:

1. There are two types of tabs:
   - **Data Storage Tab:**
     
     It shows messages as data stores are created and removed.
   
   - **Package Tab:**
     
     It shows list of queries with row count.

2. There are three types of cached tables: **permanent, cached and temporary.**

3. **Temporary tables** are used by **text** and **xml connectors** or some **transform queries.**

4. When you preview the query, the preview query never cached. But the dependent queries (queries on which previewed query is based on) will be cached.

5. If you want to flush single query right click on the query name and click on **flush table.**

6. If you want to make table permanent in the database, right click on the query name and click on **make permanent.**

7. To flush all table click on **Flush all. Flush all** only removes temporary queries.
8. When temporary query is flushed, it will still show on the screen. You need to click on 'Refresh' to remove all non existence queries

**Repository:**

Data Repository contains data connections, names of connections and brief descriptions of connections that can be shared among a number of Data Process Logic packs in a system.

By adding a connection to a repository, a user will not have to create the same connection for each Data Process Logic pack.

Data Repository allows you to create and store the connection to be shared among all Data Process Logic packages in the same machine.

**Note:** How we can add connections in a repository is explained in detail in PageNo : 66
Understanding Data Process Logic packages

All work done in Data Process Logic Studio takes place within an Data Process Logic package, a single file containing data connections, queries, and export templates. All components within an Data Process Logic package can communicate with one another, share data, and generally work as a coherent whole.

Sample package

For example, consider an inventory package (inventory.slfusion) designed to facilitate ordering new inventory. This package might include connections to an inventory and sales databases on the corporate server, a collection of queries, and multiple export templates, all of which share their data. The export templates might include a Product Inventory template that accesses information from the inventory database to detail the current inventory on hand; a Quarterly Sales by Product template that accesses information from a database view of what products have been sold; and an Inventory needs export template that uses information from the other two templates to subtract the components sold in this quarter from the current inventory and compute what needs to be ordered.
Understanding the data flow graph

The data flow graph displays the contents and hierarchy of the Data Process Logic package and allows you to see how data is moving through your project. As Data Process Logic packages grow more complex, the data flow graph becomes a crucial component for visualizing and diagnosing your data workflow.

When you open or create a new Data Process Logic package, the Document area of the screen displays the Data Process Logic package editor document. This document contains two panes: the Data Flow graph (on the top) and the Preview pane (on the bottom)
Each item in the **Data Flow** map is called as a **Node**; the grey lines - represent the **Data Flow** among the **Nodes**. As you work on the package, **Data Process Logic Studio** creates and removes nodes to reflect your changes. **Data Process Logic Studio** will attempt to find a location for newly created nodes. **Nodes** can be rearranged anywhere on the screen by clicking and dragging them within the document. The placement is strictly for visual clarity; it has no effect on your **Data Process Logic** package.
To view the results of a query within the **Data Flow** model, right-click on a **Node** and select **View Data**. You can also view data by double-clicking the query **Node**. Use the **Start row** and **Max row** sliders to control preview of the data.

Different icons represent different types of items in the package. A background indicates selected items.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="data connector" /></td>
<td>data connector (unselected)</td>
</tr>
<tr>
<td><img src="image" alt="data connector" /></td>
<td>data connector (selected)</td>
</tr>
<tr>
<td><img src="image" alt="standard query" /></td>
<td>standard query (unselected)</td>
</tr>
<tr>
<td><img src="image" alt="aggregate query" /></td>
<td>aggregate query (unselected)</td>
</tr>
<tr>
<td><img src="image" alt="transform query" /></td>
<td>transform query (unselected)</td>
</tr>
<tr>
<td><img src="image" alt="export template" /></td>
<td>export template (unselected)</td>
</tr>
</tbody>
</table>
Although **Node** placement does not affect the connections, you can use the **Nodes** and connections of the data flow map to perform certain tasks:

- **To add a new connection or query**, right-click in a blank area of the **Data Flow** map and select **New Connection** or Create [type] query from the shortcut menu.
- **To edit a query**, right-click its **Node** and select **Edit** from the shortcut menu.
- **To add a filter to a query**, right-click its **Node** and select **Filter Wizard** from the shortcut menus.
In the Filter Transformation Wizard right-click on the value in the datagrid and select either **Add to Filter** or **Exclude from filter**. Edit the filter condition as desired.

Filter Wizard to help users to create a New Transform Query with the filter condition from an existing query.

While we are applying two or more filter conditions for a query we can use logical operators **AND** or **OR**.

Click **Finish** to exit the dialog and return to main **Data Process Logic Studio** window.
- On the **Data Flow pane**, note the new Node created of the original query. This Node is a transform query containing the result of the filter condition selected in the wizard.

- To build a New Transform Query based on the column(s) a user selected on the wizard in an existing query.

- From the Data Flow pane, right click on a query and select **Select Columns Wizard** under Wizards.

- In the **Select Column Transformation Wizard**, click on the column(s) you wish to include in the new query.

- Click **Finish** to exit the dialog and return to the main **Data Process Logic Studio** window.

- On the **Data Flow pane**, note that the new **node** created off the original query. This **node** is transform query containing the columns selected in the wizard.
To edit a connection, query, right-click its Node and select Edit from the shortcut menu.

To delete a connection, query, from the Data Process Logic package, right-click its node and select Delete from the shortcut menu.

To preview the data collected by a query, double-click the desired query Node or right-click it and select View Data from the shortcut menu.

To describe the query or connection, click the desired query Node and select Describe. Write any description for
the connection or query on the Enter Description dialog and click OK.

- **To export data** using an export template, right-click the template Node and select “Run Export” from the shortcut menu.

- **To Copy the whole data flow** of the Data Process Logic package, right-click in a blank area of the Data Flow map and select Copy image to clipboard option. It will copy the whole data flow then paste it back in any other application, including Microsoft word, Microsoft paint, etc…
Installing Data Process Logic Studio

Data Process Logic Studio comes in a single Windows installer image. In most cases, the installation wizard will quickly guide you through the process without the need for any additional assistance.

Before you begin the installation, verify that you have met the system requirements (detailed on the next page). In particular, be sure that you have the appropriate components for accessing your current data stores. You can install these components before or after installing Data Process Logic Studio:

- If you wish to connect to Microsoft® Access or Excel, Microsoft® MDAC is required.
- If you wish to connect to an Oracle® database, Oracle® ODP is required.

Both components are available as free downloads from the Microsoft and Oracle sites. (Links to the downloads are posted on the Data Process Logic Studio web site, http://www.Data Process Logic.com)
Understanding system requirements

Before you begin installation, make sure that your system meets the minimum requirements.

- Microsoft® Windows Server 2003/XP/2000, Microsoft® Windows NT 4.0 SP6a or later, or Microsoft® Windows 98/ME
- 512 MB memory
- 1 GB free disk space
- Microsoft® MDAC 2.6 or later (already present on most machines to access data from most Microsoft® data sources)
Installing Data Process Logic Studio

To install Data Process Logic Studio

2. On the Welcome screen, click Next.
3. Read the **License Agreement** dialog, select **I Agree** and then click **Next**.
4. On the **Customer Information** screen, fill the appropriate **Name** and **Organization**, then click **Next**.
5. On the **Select Installation Folder** screen, click the **Browse** button to select the root folder for your **Data Process Logic Studio** installation.

![Select Installation Folder](image)

6. You can use the **Disk Cost** tool to help you determine the hard disk you will use for installation, based on the amount of disk space required.
7. Now it will ask for **Confirmation Installation**. Then click **Next**.
8. After selecting the appropriate installation folder, click **Next**. Wait a moment for the installer to complete the process and then click **Next** again.

![Installation Progress]

9. After Installation is completed, it will show you **Installation Complete** screen, Click **Close** to exit from the exiting window.
Installation Complete

Software Labs xFusion Studio 4.1 has been successfully installed.
Click “Close” to exit.
Understanding the Disk Cost tool

The Disk Cost tool analyzes the amount of available storage space and compares it to the amount required for your Data Process Logic Studio installation. Note that the tool looks at the hard drives connected directly to your machine as well as any network volumes that you have mapped to your system. It does not analyze your entire network; only those volumes you have mapped.

- **Volume** identifies the attached drive, partitioned volume, or mapped network drive/volume
- **Disk Size** displays the full size of the disk (used and unused space)
- **Available** displays the amount of unused space on the drive/volume.
- **Required** is the amount of space required to install the Data Process Logic Studio Designer components you have selected.
- **Difference** reports the amount of space that will remain unused on the drive/volume after you have installed Data Process Logic Studio Designer.

**TIP:** The Difference figure allows you to plan your installation with regard to the number and size of the reports that you expect to create.
Licensing Data Process Logic Studio


To license Data Process Logic Studio

1. Run Data Process Logic Studio by double-clicking the Data Process Logic Studio icon on the desktop
   - OR -
   Go to the Windows Start menu and select All Programs > RISE Corp DataPro Logic 4.3 > License Manager.

2. Go to Help Menu and select Licenses button, A Licenses dialog appears.
3. 
4. Click the **License** button on the right-hand side and enter the key obtained from the license card or from the web.

To upgrade an evaluation license to a permanent license

You can return to the license information dialog by selecting **Licenses...** from the **Help** menu. Click the **License** button again and enter the permanent key. You have to then activate your license by clicking the **Activate** button from within **Data Process Logic Studio**. If you are unable to activate **Data Process Logic Studio** due to network or security restrictions, you can activate it manually by contacting **support@risecorp.com**
Data Process Logic Engine uses data connections to acquire the information for manipulation and export. Data connections are software components used to communicate directly with a data source (such as an Oracle or Microsoft SQL Server database).

A data connection includes information about the server that hosts the data source, the location, appropriate authentication methods, and any necessary configuration. Data Process Logic Studio provides a wizard that will guide you through the creation of data connections for each connector. These dialogs are similar to the Microsoft Windows Data Link Properties dialogs.
Creating data connections

Before creating queries and export templates in Data Process Logic package, you must identify the source of the data by creating a data connection.

**Note:** You can create new data connectors for Data Process Logic Studio only within an Data Process Logic package (.slfusion).

To create a data connection (general steps)

1. If you do not already have an Data Process Logic package file, open Data Process Logic Studio and run.
- Create a new **Data Process Logic** package by going to the **File** menu and selecting **New** or by clicking on **Action: Create a new Fusion package** in the **Dynamic Help** pane.
- A new tab appears at the top of the workspace with the name of your package file or **Untitled** if you have created a new **Data Process Logic** package file.
-OR-

You can open an existing **Data Process Logic** package by going to the **File** menu and selecting **Open**. Navigate to the desired file (*.slfusion), or select **Action: Open an existing Fusion package** from **Dynamic Help** pane

2. Go to the **Dynamic Help** pane (top of left-hand pane) and select **Action: Create a connection to your data**
Go to the Data pane (in the bottom of the left-hand pane), right-click on the Connections icon, and select New Connection.

-OR-

Go to the Document Area (top of right-hand), right-click on the blank space, and select New Connections.

3. When the Create Connection dialog appears, you will be prompted to Select a connection type from a list that displays all the connection providers available on your system. Click to select the type of data connector you wish to use.

Data Process Logic Engine ships with a variety of pre-packaged connectors, including the following:

- Microsoft Dynamic CRM Data Connector
- Quick Books Data Connector
- Sage ACT! Data Connector
- SAP Business one Data Connector
- SalesForce Data Connector
- Peach Tree Data Connector
- SAP Data Connector
WebDB Data Connector
MySQL® Data Connector
ODBC Data Connector
OLE DB Data Connector
Oracle® Data Connector
PostgreSQL Data Connector
SQL Server® Data Connector
Microsoft® Access Data Connector
Microsoft® Excel Data Connector
Text Data Connector
XML Data Connector
Windows Management and Instrumentation Data Connector

For more information on how to set up a specific connector type, see in the Connector Reference Guide.

1. By default, Data Process Logic Studio creates connections with the name New Connection. Once you have defined a connection, you should give it a more meaningful name. To rename the connection, slowly click on the New Connection name and press F2, or right-click on New Connection, select Rename and then edit the connection name.
Note: Notice that the tasks available in the Dynamic Help pane change as your Data Process Logic package grows.

2. After you create the desired connection(s), you will want to proceed by creating queries, as described in the chapter Data selection: query
Creating Data Repository

Data Repository contains data connections, names of connections and brief descriptions of connections that can be shared among a number of Data Process Logic packs in a system. It provides not only ease of use and convenience when a user need to use the same connections for a different Data Process Logic pack, but also connection data security because the connection data will not be exposed when a Data Process Logic pack is shared and sent over the network. By adding a connection to a repository, a user will not have to create the same connection for each Data Process Logic pack.

Note: Data Repository allows you to create and store the connection to be shared among all Data Process Logic packages in the same machine.
To add a data connection to the data repository:

1. Under **Tools**, select **Repository...**

2. In **Connection Repository** dialog, right click and select **Add Connection**.
3. Select the desired connection string from Create Connection dialog.
### Create Connection

**Select a connection type**

The following list contains all connection providers found on your system:

<table>
<thead>
<tr>
<th>Connection Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesforce data connector</td>
</tr>
<tr>
<td>SAP Business One Data Connector</td>
</tr>
<tr>
<td>WebDB Data Connector</td>
</tr>
<tr>
<td>Sage ACTI Data Connector</td>
</tr>
<tr>
<td>SAP Data Connector</td>
</tr>
<tr>
<td>Salesforce Data Connector</td>
</tr>
<tr>
<td>QuickBooks Data Connector</td>
</tr>
<tr>
<td>Microsoft Dynamics CRM Data Connector</td>
</tr>
</tbody>
</table>
4. Once the connection has been created successfully, a connection named **New Connection** will appear on the **Connection Repository**.

![Connection Repository](image-url)
5. You can rename the connection or change (add) the description of the **New Connection**.

6. You can also edit or remove any existing connection by right-clicking on **New Connection** from the menus.
7. To use a connection from a repository, right click on **Connections** from **Data** pane and select **New Linked Connection** from the menu items. Select the desired Connection ID from **Connection Repository** and click **OK**.
Creating Virtual Connection

It is very normal to have a number of data connections, for example, to a test system and a final system, etc. in an Data Process Logic package. Virtual Connection provides a flexible way to switch among these many connections. You can create as many data connections as needed and create the virtual connection from one of the connection. Then as needed, you can edit the virtual connection and change the parent of the virtual connection to any connections created in the Data Process Logic package.

Note: Virtual connection allows you to switch between the data connections in Data Process Logic package.

To create a virtual connection:

1. Right click on the data connection created and select Create virtual from shortcut menu in data pane.

   OR

   You can right click on any connection node in document area and select Create virtual from shortcut menu.
2. The new virtual connection will be created off from the original connection. Rename the new virtual connection if desired.
3. Create a New Standard Query using the virtual connection.

4. You can switch between connections by right-clicking on the virtual connection created earlier from Data pane and select Edit. The Edit Virtual Connection dialog will show
up. Here you can click on the drop-down box and change the connection as needed.

![Edit Virtual Connection](image)

5. Once the virtual connection is edited, the data flow will be moved to show that the data source is now **TestDB**.

![Data Flow](image)
**Associate Data Models**

Using the Associate Data Model option on any of the connection node, we can create relational standard queries and get the data from data models that are related to the selected connection.

To associate data models:

1. **Create** a connection. Create a standard query and save it.

2. **Create a data model** for the standard query in the desired directory and save the fusion pack.

3. Close the fusion pack and open it again.

4. Right click on the **Connection** node in the data flow pane and select **Associate Data Model** option from the shortcut menu. Select the desired Application as shown below.
In order to associate the data model for a **repository connection**, select Repository under Tools menu, right click on the desired connection from the **Connection Repository** window and select **Associate Data Model** option and set the desired Application as shown below.
5. Once the data model is associated with the application, then create a standard query to view the data from the data model. The Data Model tab in the right side of the standard query will display all the data models associated with that connection for the specified application.
Drag any data model to the control pane and click Preview button to see its data.
Queries within the **Data Process Logic Studio** environment encapsulate a process that produces tabular data. **Data Process Logic Studio** queries should not be confused with SQL queries or functions that return tables. SQL queries or functions can be encapsulated into a query but are generally only building blocks of a more complex **Data Process Logic** query.

**Data Process Logic Studio** provides three types of queries:

a. standard
b. aggregate
c. transform

Each query type uses a different process to fetch or construct data. However, once constructed, all queries are treated in the same way, using the procedures described in this help file.
Organizing queries

The Queries folder in the Data pane can contain a number of queries. These queries can be designating in particular categories for organization by right-clicking on Queries folder and select New Category from menus. To add a query into a category, click and drag an individual query to the desired category folder.

- To organize queries, click and drag an individual query to the desired category folder.

You can use standard Windows procedures within Queries folder:

- **To edit a query**, right-click on the desired query and select **Edit** from the list. This will open the query in another tab for editing.
- **To add description**, right-click on the desired query and select **Description** from the list. Enter the description in the text box of the **Enter Description** dialog and click **OK**.
- **To delete a query**, right-click on the query and select **Delete** from the list.
- **To duplicate a query**, right-click on the query and select **Duplicate** from the list. Duplicating creates an exact copy of the query. Once you create duplicate of the query, you can adjust various parameters,
providing an easy way to create similar but not identical queries.

- **To rename a query**, right-click on the query and select **Rename** from the list. Type in the new name of the query.

- **To quick export a query**, right-click on the query and select **Quick Export** from the list.

- **To add a filter to a query**, right-click on its node and select **Filter Wizard** from the shortcut menus. It will create a new transform query.

- **To build a new transform query** from the data in an existing query, right-click on the existing query’s **Node** and click on **Select Columns Wizard** from the shortcut menus.
Creating Standard Queries

To create a standard query, you must already have created at least one data connection within Data Process Logic package. Once the data connection is established, you can begin creating a standard query in one of three ways:

- Right-click on the Queries folder from the Data pane and select New Standard Query from the shortcut menu.
- Right-click on a blank space in the background of the Document Area and select New Standard Query.
- You can also create a standard query by clicking on Action: Create a new standard query from Dynamic Help pane.

Once created, standard queries can be refined in either design (drag-and-drop) or advanced (native language) mode.

NOTE: Before switching from design to advanced mode, it is advised you save the query. (From the File menu, select Save [query name] where [query name] is the name of the current query). Once you are in advanced mode, clicking to design mode will clear the current query. You can only reload the query by clicking on its name in the Data pane if you saved the query before switching to advanced mode.
Creating Standard Queries in Design Mode

Design mode is the default for standard queries and the only mode available for aggregation queries. The design mode presents a visual abstraction of tables and joins. When you work with a query in design mode, the right side of the query editor lists the tables, views, and stored procedures found in the data connection that is serving as the source for the query. The query editor's other panes reflect your work as you build the query.
The **Content pane** (top left side of the editor) displays the elements of your query as you build it.

The **Native language pane** (center-right) displays the query you are designing in the native-language of the data connection.

The **Preview pane** (lower-right) displays a sample of the data being returned by your query.

The graphic shows the query editor as it appears when first opening a standard query in design mode. Except for the area listing elements found in the source data connection, the editor's panes are blank because no elements have yet been selected or refined.

To create a query in design mode

- Choose the appropriate connection from the **Connections** drop-down in the toolbar.

- Select the tables or procedures that contain the desired data by clicking and dragging the desired object to the content pane.

**Note:** If a table contains Phantom in the title and has a gold title bar, the table no longer exists due to a change in the connection or the underlying database. If you are certain the query is correct, phantom tables can still be used and joined across. If you delete a phantom table, however, it cannot be retrieved again.
- **Data Process Logic Studio** caches the table and column metadata for all connections used inside the **Data Process Logic** package to avoid extraneous queries to the database.

- To see how recent cached date is, hover over a table name in the **Tables** pane.

- To get new table metadata, use the **Refresh** tool on the toolbar to update it.

- To refresh column metadata, right-click on a table in the **Tables** pane and select **Refresh Columns** from the shortcut menu. If the table is already being used, it will not update automatically to prevent loss of the current query. If you wish to use the new column metadata, close the table and drag it over again.

- If you are using multiple tables, **joins** can be created by dragging a column from one table onto a column of another table, automatically creating an inner join where the two columns are equal.

  - **To change the join type, condition, or relation**, right-click on the **Join line** and select the **Join Type**, **Join Condition**, or **Join Relation** from the shortcut menu.

  - **To change the join order**, the relationship between joins can be set from the same shortcut menu. Use the query in the **Native language** pane to determine what the join order is.
- Select the desired fields/columns from the table, using the small dialog that appears when the table is positioned in the content pane.

- To include all values from a particular field in your query, click the **Check-box** to the left of the field name.
- To include all values of all fields, click the * in the lower left corner.
- Right mouse click on title bar of the table you will find
  - **Sort Column:** Select the required order, using the small dialog that appear.

- **Show Column:** Select the required option, using the small dialog that appear to show relative columns.
Refine the data selection from each field by creating filters, groups, and order conditions. Right-click on the desired field and select Alias, Add Filter, Add to Group, or Add to Order from the shortcut menu.

- **Add to Order** opens a secondary menu allowing you to choose whether the contents of the field should be presented in Ascending or Descending order.
- **Alias** allows you to specify a new name over an existing column name.
- **Add to Group** adds the selected field to a group list maintained for this query. If you add more than one order or group condition, all will appear in a list on the **Query Properties** dialog's **Group Conditions** or **Order Conditions** tabs.

- **Add Filter** open **Query Properties** dialog allowing you to specify the details of the condition.

![Query Properties dialog](image)

- The first item in the condition is the specified field, in the form Table.Field.

- The second item in the condition is the operator, which is set to = (equals) by default. To choose a different operator, right-click the = and select the desired operator from the shortcut menu (choices include <, >, <=, >=, Not, Like, In, Between, and Is Null). Note that
The operator Not can be combined with any other operator.

- The final part of the condition contains one or more text boxes, depending on the operator. Click on a Text box and type the appropriate value. Some connectors will attempt to automatically format values to the correct syntax although it is still possible to enter a value that will not create a valid query. Parameters can be used by providing a parameter name within braces { }. When the query is run, Data Process Logic prompts for the parameter value, and then substitutes that value in place of the parameter. If you are having difficulty getting a parameter value to work, consult the type formatting documentation for that specific database.

- Continue adding filters until you have achieved the desired query. You can also adjust filter relationships and use toolbar icons to check the progress of your query-building:
  - To change the filter order, click and drag the filter list item to the appropriate position on its Filter Conditions tab.
  - To remove a filter item, click the red X to its right.
  - To determine how much data your query will return, click the Row count icon on the toolbar.
**Note:** On filtering, when you enter a value for date field, the syntax is different for **MS SQL** versus **MS Access**

If you use local data store (**Access database**) you have to put # to that value at starting and end.

For example:

```sql
SELECT [inv1].[DocDate], [oinv].[CardCode], [oinv].[CardName] FROM ( [oinv] INNER JOIN [inv1] ON [oinv].[DocEntry] = [inv1].[DocEntry]) WHERE [inv1].[DocDate] = #1/20/2006 12:00:00 AM#
```

If you select SQL Server as data store you have to put the value within the quotes (’ ’)

For example:

```sql
SELECT [inv1].[DocDate], [oinv].[CardCode], [oinv].[CardName] FROM ( [oinv] INNER JOIN [inv1] ON [oinv].[DocEntry] = [inv1].[DocEntry]) WHERE [inv1].[DocDate] = '1/20/2006 12:00:00 AM'
```

- You can preview the results of your query by clicking the **Preview** icon on the toolbar. The data returned by your query appears in the preview pane.
To adjust the amount of data returned in the **Preview** pane, use the **Start row** and **Max row** sliders above the **Preview** pane.

For other adjustments and checks, use the toolbar icons:
- **Row Count:**
  number of rows of data the full query will return

- **Conditions:**
  edit the query’s conditions

- **Parameters:**
  edit the query’s parameters

- **Smart Join:**
  **Data Process Logic** identifies and creates appropriate joins between the tables based on matching column names

- **NOTE:** When you finish working on a query in design mode and want to perform more edits in advanced mode, save the query before switching from design to advanced mode. This creates a save point to which you can revert if you are not satisfied with the changes you make in advanced mode. Once you are in advanced mode, clicking to design mode clears the current query. You can only reload the query by clicking on its name in the **Data** pane if you saved the query before switching to advanced mode.
Creating Queries in Advanced Mode

In advanced mode, queries are represented in the native language of the data connection. SQL is the language for most databases; however, the language can vary by the connector. Some connectors such as XML and text do not have a native query language and therefore do not support using advanced mode queries.
Create a query in advanced mode

1. Create a New Standard Query. You may begin to define conditions of the query in design mode, but it is not required.

   NOTE: When you finish working on a query in design mode and want to perform more edits in advanced mode, it is advised you save the query before switching from design to advanced mode. This creates a save point to which you can revert if you are not satisfied with the changes you make in advanced mode. Once you are in advanced mode, clicking to design mode clears the current query. You can only reload the query by clicking on its name in the Data pane if you saved the query before switching to advanced mode.

2. Click the Advanced mode drop-down list from the toolbar. The Content area of the editor now allows you to type the query text in SQL or other native query languages. For reference on SQL or other query languages, consult the documentation for that specific database.

3. The Connections drop-down list in the toolbar displays the connection currently being used. You can freely change connections while editing the query.

   Note: Query syntax is generally not compatible between different connectors, and switching to a connection that does not support advanced queries will clear your current query and convert it to a design query.
Using the Expression Editor

The **Data Process Logic Studio Expression Editor** provides a way to manipulate data into new forms. For any given connector, the expression syntax and functions available are those native to the connector used in the query. (For example, when querying from Oracle, you may only use Oracle-native syntax and functions.) The **Data Process Logic Studio** expression syntax is used with text and XML connectors and in transform queries. Cover specific functions and the **Data Process Logic** expression syntax used to combine them. You can type expressions directly when working in Advanced mode, but must use the **Expression Editor** within Design (Relational) mode.

To use the Expression Editor

1. Open the desired query in the document window by double-clicking the query in the Data pane or right-clicking the query in Data Flow and selecting Edit from the shortcut menu.
2. Open the **Expression Editor** by clicking the "+" sign in the lower right corner of the table you wish to use.

![Table with check boxes for different options]

3. The **Expression Editor** appears as the dialog shown below.

![Expression Editor dialog]

---

*Data Process Logic Studio* User Guide 101
4. Drag and drop the functions to the text box at the bottom of the
editor to construct the desired expression. Type in the
appropriate Data Process Logic expression syntax or native
syntax to connect the drag-and-drop elements.
-OR-
Type all expression elements, values, and syntax directly into
the text box at the bottom of the editor.

5. When the expression is complete, click OK to return query
editor screen.

6. When you are satisfied with the query, go to the File menu
and select Save [query name] where [query name] is the
actual name of your query.

7. If you want to know the details of the table, go to the
Tables pane and right-click on the desired table and select
the option Table Details.
Here you can view the table structure. In **Relations** tab it will show Foreign column, Foreign table and Primary column.
Creating Data Model

A Data Model deals directly with the meta data from a standard query. In their turn, Data Models can become the data sources for aggregate and transform queries.

1. **Create** a standard query and save the query.
2. Once the standard query is created, you can begin creating a Data Model.

   - Right mouse click on **standard Query** node and select **Create Data Model** option from shortcut menu as shown below.

![Create Data Model example](image-url)
After selecting the Create data model from shortcut menu, Create Data Model wizard got opened as shown below.

3. Specify the required parameters.

- **FileName**: By default it will display the path to the application directory, but we can change the file name by clicking on (...) browse button.
- **Data Model Name** - By default it will take the standard query name and if we want to change the data model name we can change it.
- Give valid values to the remaining fields (Description, Application and category) in the create data model wizard.
- **Description** field meant by, description regarding the data model.
- **Application** field meant by, the type of Application like connection type (SAP, SAP B1 ... etc) by giving the application name it will put the related data model under this name of the application.
- **Category** field meant by, type of category of data model like (Inventory, BusinessPartners ... etc) by giving the category name it will put the related data models under this name of the category.

4. Click on **OK** button it will create the data model as a file with the file extension **.dm**.
Understanding aggregate queries

Aggregate queries allow joins across two or more standard queries or transform queries within Data Process Logic Studio. You most often use aggregate queries to combine data from two standard queries or transform queries using different connectors.

NOTE: For best performance, use joins within standard queries or transform queries instead of aggregate queries whenever possible. Using aggregate queries on large volumes of data can degrade performance.
Creating aggregate queries

Design mode is the only mode available for aggregate queries. The Design mode presents a visual abstraction of tables and joins. When you work with a query in Design mode, the right side of the query editor lists all the Data Process Logic Studio queries defined in the current package. The query editor's other panes reflect your work as you build the query and will be blank when you first begin to create your aggregate query.

The Content pane (top left side of the editor) displays the elements of your query as you build it.

The Native language pane (center-right) displays the query in SQL (Structure Query Language). This is used only for reference and is never submitted to any connector.

The Preview pane (lower-right) displays a selection of the data returned by your query.

To create an aggregate query

1. Before creating an aggregate query, you must already have created at least two other queries (standard, aggregate, or transform) within your Data Process Logic package.

2. Create a blank aggregate query:
• Right-click on the **Queries** folder in the **Data** pane (bottom left pane) and select **New Aggregate Query** from the shortcut menu.

  -OR-

• Right-click on a blank space in the background of the **Document area** and select **New Aggregate Query**.

3. Select the tables and views that contain the desired data by clicking and dragging the desired table or view to the **Content** pane. If you do not see the tables or columns you want, the metadata may have gone out of date. **Data Process Logic Studio** works with cached table and column metadata to enhance performance by reducing extraneous queries to the database.

  - To check the age of the metadata, however the mouse over a table in the **Tables** pane.
  - To refresh table metadata, use the **Refresh** icon on the toolbar.
  - To refresh column metadata, right-click on a table in the **Tables** pane and select **Refresh Columns**. If the table is already being used, it will not update automatically to prevent loss of the current query. If you wish to use the new column metadata, close the table and drag it over again.
Note: If a table contains Phantom in the title and has a gold title bar, the table no longer exists due to a change in the connection or the underlying database. If you are certain the query is correct, you can still use the Phantom tables and create joins across them. If you delete a phantom table, however, you cannot retrieve it.

4. If you are using multiple tables, create joins between tables by dragging a column from one table onto a column of another table, automatically creating an inner join where the two columns are equal.

- To change the join type or condition, right-click on the Join line and select the Join Type, Join Condition, or Join Relation from the shortcut menu.

- To change the join order, the relationship between joins can be set from the same shortcut menu. Use the query in the native language pane to determine what the join order is.
5. Select the desired fields/columns from the table, using the small window that appears when the table is positioned in the Content pane.
   - To include all values from a particular field in your query, click the check-box to the left of the field name.
   - To include all values of all fields, click * in the lower left corner.

6. In case that the standard queries used in an aggregate query contains parameter(s). The parameter will be passed on automatically. However, this will have implication on additional queries using this aggregate query that the parameter must be specified with an absolute query and column name, for example, DataFormatted, parameter.

7. When the selections are complete, save the query from the File menu, select Save [query name] where [query name] is the name of the current query.
Understanding transform queries

Transform queries use Data Process Logic Studio expressions to manipulate and transform existing queries (standard, aggregate, or transform) to produce new data and new data formats. Transform queries are created using the transform query editor, which uses the built-in Data Process Logic Studio functions and logic to transform the data in the underlying tables.

Data Process Logic package always presents the final value of a transform query is as a table, regardless of the dimension of the source expression.

- If you create a transform query, which uses or returns a scalar (individual) value, the transform query produces a table with a single column and single row containing that value.
- If you create a transform query which uses or returns a vector (one-dimensional series) value, the transform query produces a table with a single column displaying that vector.
- If you use a transform query within another transform query, it will always be treated as a table and will return a table as the result.
Sample transform queries

To understand transform queries, consider the following examples. Query1 and Query2 form the basis for all of the sample queries.

Query 1

<table>
<thead>
<tr>
<th>Column1</th>
<th>Column2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>e</td>
</tr>
<tr>
<td>c</td>
<td>f</td>
</tr>
</tbody>
</table>

Query2

<table>
<thead>
<tr>
<th>Column1</th>
<th>Column2</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>dd</td>
</tr>
<tr>
<td>bb</td>
<td>ee</td>
</tr>
<tr>
<td>cc</td>
<td>ff</td>
</tr>
</tbody>
</table>
Combining columns from several queries into a single table

The **Select** function combines multiple queries into a single table. **Select** takes an arbitrary number of parameters of type Vector. Applying **Select** to the sample queries above produces results as shown below.

**Sample Query 1:**

Select ([Query1.Column2], [Query2.Column1])

**Sample Result 1:**

<table>
<thead>
<tr>
<th>Column1</th>
<th>Column2</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>aa</td>
</tr>
<tr>
<td>e</td>
<td>bb</td>
</tr>
<tr>
<td>f</td>
<td>cc</td>
</tr>
</tbody>
</table>

**Sample Query 2:**

Select ([Query2.Column1], [Query1.Column2], Array(1,2,3))
**Sample Result 2:**

<table>
<thead>
<tr>
<th>Column1</th>
<th>Column2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>bb</td>
<td>e</td>
<td>2</td>
</tr>
<tr>
<td>cc</td>
<td>f</td>
<td>3</td>
</tr>
</tbody>
</table>

**Sample Query 3:**

```
Select ([Query1.Column1] as C1, [Query2.Column2] as C2)
```

**Sample Result 4:**

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>dd</td>
</tr>
<tr>
<td>b</td>
<td>ee</td>
</tr>
<tr>
<td>c</td>
<td>ff</td>
</tr>
</tbody>
</table>

Note the keyword “as” was used in this example. "as" specifies a name for the column, in this case “C1” and “C2”. You can
use “as” after any Vector type, but only in certain circumstances.

**Filtering rows from queries**

The Filter function allows you to filter rows from a query by specifying filter conditions.

**Sample Query 4:**
Filter ([[Query1]], [Query1.Column1] <> “a”)

**Sample Result 5:**

<table>
<thead>
<tr>
<th>Column1</th>
<th>Column2</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>e</td>
</tr>
<tr>
<td>c</td>
<td>f</td>
</tr>
</tbody>
</table>

**Sample Query 5:**
Filter ([[Query1]], ([Query1.Column1] = “a”) or ([Query1.Column2] = “f”))

**Sample Result 5:**

<table>
<thead>
<tr>
<th>Column1</th>
<th>Column2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>f</td>
</tr>
</tbody>
</table>
Creating transform queries

Transform queries manipulate existing standard or aggregate queries using Data Process Logic studio expressions to produce data. The results of a transform query is always presented as a table regardless of whether the dimension of the input. (For more information, see Understanding transform queries.)

To create a transform query

1. Right-click on the Queries folder in the Data pane (bottom left pane) and select New Transform Query from the shortcut menu.
   -OR
   Right-click on a blank space in the background of the Document area and select New Transform Query.

2. The upper right side of the query editor displays a list of the queries in the current Data Process Logic package as well as a categorized list of the available Data Process Logic Studio functions.

   **TIP:** Hover the mouse over any function in the query editor to see a tooltip displaying the function’s purpose, syntax, and required parameters.
- Drag and drop the desired tables, fields, and functions to the text box at the bottom of the editor to construct the desired expression. Type in the appropriate Data Process Logic Studio expression syntax to connect the drag-and-dropped elements.

- You can also type all expression elements, values, and syntax directly into the Document area.

3. To preview the results of your query, click the Preview icon on the toolbar. The data returned by your query appears in the Preview pane. To adjust the amount of data returned in the preview, use the Start row and Max row sliders above the Preview pane.

4. When you are satisfied with the query, go to the File menu and select Save [query name] where [query name] is the actual name of your query.
Validation Rules

Validation Rules are a set of queries (standard/advanced/aggregate/transform) defined to prevent bad data uploads. Define a validation rule to run related validation queries in the specified order. If the validation query returns one or more rows, it is considered to have failed from validation perspective. For example, a query could be designed to "show all the records where customer id is missing." If the query returns any records, it implies we have bad data.

Email Configuration:

Email configuration is required to send email notification on export success or failure. Create a Mail.config file with the mail host server name, username, password information. This configuration file is read to send emails if specified in the export template. Below is the sample format for Mail.config file.
Note: For Microsoft Windows Vista or Windows Server 2008 operating system, Mail.config file has to be created under directory `c:\ProgramData\RISE\4.3\` and for any other operating system create under `C:\Documents and Settings\All Users\Application Data\RISE\4.3\`
Steps to create Validation Rule

1. Create one or more queries (standard/advanced/aggregate/transform) for validation purpose.

2. Right click on the data flow screen to find Validation rule option. You can also add a validation rule from the data panel to the left.

3. The dialog below will open when the validation rule option is selected.
4. The dropdown list will show all queries. Select the query name designed for validation. The failure message is optional. Each validation rule may have multiple validation queries.

5. Click on **Run** to execute the selected queries in the specified order. When a validation rule is run, it returns false if any of the validation query has failed. In addition, it returns a HTML text in the bottom pane with the failure message and failed records.

6. Click on **Save** to save the rule.

7. In the left hand side **Data panel**, right-click **Export Templates** and select **New Template** from the shortcut menu.

8. In the **Data Export** wizard, you must first create a **Destination Connection**. From the list of available connectors, click to select the type of data connector you wish to use. Destination connections are set up in much the same way as source data connections.

9. Follow the wizard to find the validate/Notify task.
10. Select the dropdown list to display all the validation rule defined in the fusion pack. This is optional. By default no rule is selected.

11. Optionally you may specify the validation output filename. \%FPDIR\% stands for fusion pack directory. This output file contains the result of the validation rule in HTML. This HTML file will be emailed as an attachment.

12. In addition, you can optionally specify the email address to send the result of the export. Click on Test Email to check if the email configuration.

13. Click on Next.
14. Click on Run export to start the export. If the validation rule is selected it will be run and if the rule returns false the export fails. A success or failure email will be sent if the notification was specified.
Data Export: Templates

Data Process Logic Export templates allow you to create reusable formats so that you can quickly and easily draw data from one source, manipulate it as desired, and output it to another data format. For example, you could create a template for your month-end sales reports that extracts data from the corporate Oracle database, separates and summarizes sales by region and sales person, then outputs the result to a Microsoft® Excel spreadsheet. Once the template is created, you can easily schedule it using the export command line utility packaged with Data Process Logic Studio.

Export templates draw their data from existing queries within an Data Process Logic package that filter and manipulate the information extracted from the data connection before sending it to the data destination. Templates also define how source data columns map to the destination table, and how the destination table should be created.
To create an export template

1. Open an existing Data Process Logic package or create a new one.

2. In the Data pane, right-click Export Templates and select New Template from the shortcut menu.

   - In the Data Export wizard, you must first create a destination connection from Set destination connection dialog for your exported data. From the list of available connectors, click to select the type of data connector you wish to use. Destination connections are set up in much the same way as source data connections. Consult the following pages for details on setting up each specific connector. Data Process Logic Studio ships with a variety of pre-packaged connectors.

3. On the Define table mappings dialog, click Add new mapping to display a list of the existing queries in the package. Click to select the query containing the desired data.

4. If the destination for export contains identifiable tables, Data Process Logic lists them and the option to Specify... a new or existing table to contain the exported data. If the destination does not contain identifiable tables, only the Specify... option is displayed. Click to choose any listed table or choose Specify... to type the name of a table. If you Specify... a table that does not exist in the destination file,
Data Process Logic Studio attempts to create the new table in the file.

5. If you select an existing table, Data Process Logic displays the fields in the source and destination tables, with arrows marking any mapping that it detects. To create additional mappings, drag fields from the source query on the left to the destination table on the right (scroll bars appear on both tables if required).

6. Click Next when you have created all desired mappings.
7. If you specify a new or existing table, or after you have identified the mappings in an existing destination table, Data Process Logic presents options for handling the export as shown below. Make your selections and click Next.

When Data Process Logic Studio returns you to the Define table mappings dialog, you can create additional mappings between the source and destination files. When you have created all desired mappings, click Next.
In **Validation and Notification** dialogue, select the dropdown list to display all the validation rules defined in the fusion pack. This is optional. By default no rule is selected.

Optionally you may specify the validation output filename. 
%FPDIR% stands for fusion pack directory. This output file contains the result of the validation rule in HTML. This HTML file will be emailed as an attachment.

In addition, you can optionally specify the email address to send the result of the export. Click on **Test Email** to check if the email configuration.
Use the **Summary** dialog to verify your export template setup. To make changes, click **Back**. When all choices are correct, click **Run Export** to export the data. Select **Finish** when the export is completed.
Setting up Access destination connections

Setting up a Microsoft® Access destination connection simply requires you to identify the desired .mdb file and supply the appropriate security information (if any).

To set up a Microsoft Access destination connection

1. From the Set destination connection dialog of the Data Export wizard, select MS Access Data Connection.

2. On the Set output file dialog, type in the complete path and Filename for the desired output file or click File... to browse to the desired location. If the file has security enabled, fill in the Username and Password in the appropriate fields.

3. Click Test Connection to have Data Process Logic attempt to connect to the specified database and report back on its success.

4. Click Next to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up Excel destination connections

Setting up a Microsoft® Excel connection simply requires you to identify the desired .xls file.

To set up a Microsoft Excel destination connection

1. From the Set destination connection dialog of the Data Export wizard, select MS Excel Data Connector.

2. On the Filename/URL field from the Connect to an MS Excel spreadsheet dialog, type in the full relative path to the file containing the desired data. You can also click on File... to browse to the desired file. If you type in a filename that does not exist, Data Process Logic will create the new file.
   -OR-
   Select the checkbox labeled Edit connection string to type in a connection string directly.

3. Click Test Connection to have Data Process Logic attempt to connect to the specified file and report back on its success.

4. Click Next to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up MySQL destination connections

Setting up a MySQL connection requires you to identify the desired server, database, and port number, then supply any appropriate security information.

To set up a MySQL destination connection

1. From the Set destination connection dialog of the Data Export wizard, select MySQL Data Connector.

2. Provide the appropriate information to identify your MySQL data source on the Connect to MySQL Server dialog.
   - In the Server field, select the desired server from the drop-down list.
   - Fill in the appropriate security information in the Username and Password fields.
   - Select the desired Database from the Database drop-down list. Data Process Logic Studio populates this list based on the Server you have selected above. Type in the appropriate port number in the Port No field if you are not using the default port.
   - You may also select the checkbox labeled Edit connection string to type in a connection string directly.
3. Click **Test Connection** to have **Data Process Logic** attempt to connect to the specified database and report back on its success.

4. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up OLE DB destination connections

Selecting the OLE DB connection from within Data Process Logic Studio allows you to use Microsoft's Data Link dialog to describe your connection. The specific forms of OLE DB providers accessible through this dialog vary according to the setup of your individual computer.

General instructions on creating an OLE DB connector are provided below. For more information on using specific versions of this provider, press F1 to access the Microsoft® Windows help system on your computer.

To set up an OLE DB destination connection

1. From the Set destination connection dialog of the Data Export wizard, select OLE DB Data Connector.

2. The Data Link Properties dialog opens to the Provider tab, displaying a list of all available OLE DB providers available on your system. Choose the desired provider and click Next.

3. The dialog moves to the Connections tab. If you have already set up the data source as a DSN connection in the ODBC Data Source Administrator (in the Windows Control Pane), you can select it from the drop-down list labeled Use data source name. Otherwise, select Use connection string.
and click the **Build** button to create a new **DSN**. Enter the appropriate security information.

4. Click **Test Connection** to make sure that **Data Process Logic** can communicate with the data source.

5. If desired, click the **Advanced** tab to adjust security-related settings such as the **Network settings**, **Connection timeout**, and **Access permissions** for the database.

6. The **All** tab allows you to edit connection settings directly by choosing a specific value, clicking **Edit Value**..., and typing the desired value into the blank field.

7. Click **OK** to exit the dialog and return to the **Data Process Logic Data Export** wizard. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up Oracle destination connections

Setting up an Oracle® connection requires you to identify the database and provide any necessary security information.

To set up an Oracle destination connection

1. From the Set destination connection dialog of the Data Export wizard, select Oracle Data Connector.

2. On the Connect to an Oracle database dialog, select the appropriate provider from the Provider drop-down list.

3. Select the appropriate server from the Service Name drop-down list, then supply any necessary security information.

4. You select the Transactional, Tables, Views, and Synonyms check box if required, by default Transactional and Tables are checked.

5. Click Test connection to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

6. Click Next to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up SQL Server destination connections

Setting up an SQL connector requires you to identify the SQL server and provide any necessary security information.

To set up an SQL data connection

1. From the Set destination connection dialog of the Data Export wizard, select SQL Server Data Connector.
2. Use the Server drop-down list from Connect to a SQL Server database dialog to identify the server that hosts your SQL Server database. The drop-down list displays the currently available servers. If you do not see your server, type in the fully qualified network path to the server.
3. Under Logon credentials, choose whether you wish to use the built-in Windows authentication or to specify a specific username and password.
   - Use Windows authentication
     Click to specify that the provider request a secure (or trusted) connection to a SQL Server running on Microsoft Windows. When selected, SQL Server uses the security credentials of the user currently running Data Process Logic Studio to authenticate. The SQL Server system administrator must have previously configured the server to use Windows authentication.
- **Use the username and password**
  Click to use a supplied user name and password to authenticate your logon information to the data source. Blank passwords are acceptable in the connection string if the data source does not require a password.

4. Use the database from **Database** drop-down list to select the file that you want to access.

5. Click **Advanced Options** to type in a connection string directly in the **Edit connection string** check box.

6. **Query Time Out** allows you to control how long Data Process Logic will try to fulfill the query before returning a timeout error.

7. Click **Test connection** to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

8. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up Text destination connections

You can use a simple text file as the output destination for your data.

To set up a text connection

1. From the **Set destination connection** dialog of the **Data Export** wizard, select **Text Data Connector**.

2. On the **Set output file** dialog, type in the complete path of the Filename for the desired output file or click **File...** to browse to the desired location. If you use an existing text file, **Data Process Logic Studio** will overwrite the data in the file. If you type in a filename that does not exist, **Data Process Logic** will create the new file.
   - Type any desired **End of row text**.
   - Select the desired **Delimiters** format.
   - Check on **Column names as first row**, if required.
   - Click **Next** to move to the next dialog in the wizard.

3. On the **Define table mappings** dialog, click **Add new mapping** to move to the next dialog in the wizard.

4. On the **Select data source** dialog, select the queries containing the data you wish to export, then select the names of the tables containing the information and click **Next** on the **Select update option** dialog.
5. Click **Next** to continue and set up the table mappings between the source and destination files.
Setting up XML destination connections

Setting up an XML connector requires you to identify the .xml file that will hold the data.

To set up an XML data connection

1. From the **Set destination connection** dialog of the **Data Export** wizard, select **XML Data Connection**.

2. On the **Select output file name** dialog, type in the complete path for the desired output file or click **File...** to browse to the desired location. If you use an existing xml file, **Data Process Logic** will overwrite the data in the file. If you type in a filename that does not exist, **Data Process Logic** will create the new file.

3. Click **Test connection** to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

4. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up SAP Business One - Business Object destination connection

Setting up an **SAP Business One - Business Objects** connection requires you to identify the desired Server name, company name, company DB, and any necessary security information.

To set up an SAP Business One-Business Object data connector

1. From the **Set destination connection** dialog of the **Data Export** wizard, select **SAP Business One-Business Object** connector.

2. Provide all necessary information such as **Version**, **Server**, **Company Name**, **Username**, and **Password**.

3. Select the appropriate **Language** which ever is desired.

4. Click on the **Advanced Option** and a dialog box appears.

5. Select the **Server Type** whichever is the backend of the SBO and either use the windows authentication or the SQL authentication mode. Click **Ok**.

6. Click Test connection to attempt a connection to the specified data source. If the connection fails, ensure that
the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

7. Click **Next** to select **Business Objects** from the list.

8. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up SAP XML IDoc destination connection

Setting up an **SAP XML IDoc Connector** connection requires you to give Schema File Name and Output Directory Name.

To set up an SAP XML IDoc connector

1. From the **Set destination connection** dialog of the **Data Export** wizard, select **SAP XML IDoc** connector.

2. On the **Schema File** field type the full path to the XML file Contain the desired data. To browse the file you can also click on **File**...

3. Provide **Output Directory** Name or ,You can also browse for folder by click on ... 

4. Click **Test connection** to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

5. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up SAP SII destination connection

Setting up an SAP SII Connector connection requires you to give Schema File Name and Output Directory Name.

To set up an SAP SII data connector

1. From the Set destination connection dialog of the Data Export wizard, select SAP SII data connector.

2. On the Schema field type the full path to the file contain the desired data. To browse the file you can also click on File…

3. Provide Output Directory Name or , You can also browse for folder by click on …

4. Click Test connection to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

5. Click Next to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up SAP destination connection

Setting up an SAP connection requires you to identify the desired server and provide any necessary security information.

To set up an SAP data connector:

1. From the Set destination connection dialog of the Data Export wizard, select SAP data connector.
2. Provide all necessary security information such as Username, Password, Client, Language, Application Server Host, and System Number.
3. You can edit the connection string by choosing the Edit connection string box and modify the connection string as needed.
4. Choose RFC/BAPI from dropdown list.
5. Click Test connection to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.
6. Click Next to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Setting up WebDB Server destination connection

Setting up a WebDB Connection simply requires you to identify the desired URL and Data Process Logic Studio Pack, and provide any necessary security information.

To set up an WebDB Data connector

1. From the Set destination connection dialog of the Data Export wizard, select WebDB Data connector.

2. On the URL field, type the URL of the WebDB Server.

3. On the Fusion Pack field, enter the name of the Data Process Logic Studio pack you wish to create a connection to.

4. Note that this Data Process Logic pack must be copied to the FusionPacks directory under WebDB Server’s installed directory.

5. Enter the Username and Password to authenticate your connection to WebDB Server on the Username and Password Fields.

6. Click Advanced Option to add authentication setting to the connection.
7. Select the appropriate **Authentication** to the authentication mode set on the virtual directory on Internet Information Services (IIS).

8. Enter **Username** and **Password** if needed.

9. Enter the **Proxy** if there is any.

10. By default, the **Batch size** is set to 500. However you can modify the batch size as needed.

11. Click **Test connection** to attempt a connection to the specified data source. If the connection fails, ensure that the settings are correct. For example, spelling errors and case sensitivity can cause failed connections.

12. Click **Next** to move to the next dialog in the wizard and set up the table mappings between the source and destination files.
Exporting data: running an export template

Creating an export template defines how Data Process Logic will export the data, but the actual export does not take place until you run the template.

To export data

1. Create an Export template or open an Data Process Logic package containing an existing Export template.

2. From the Data pane, right-click the desired template and select Run from the shortcut menu.
   - OR-
   From the Document area, right-click on a template Node and click Run Export.
   - OR-
   From the Document area, double-click a template Node.

3. When the Summary dialog appears, click Run Export.

4. When the summary says the export is completed, click Finish to return to the main Data Process Logic window.
Creating Quick Export

Quick Export is another feature that enables an export to Excel, XML and Text in a single click.

**Note:** Quick Export allows you to export data in a single click.

To create a quick export:

1. Right click on the data query created and select **Quick Export**.
2. Choose the destination connector type from the drop-down list box.

3. Modify the filename and location as desired.
4. Once satisfied, click on Run Export icon or press F5 from keyboard.

5. Once the export is completed, you can click Folder icon or press F6 from keyboard to open and view the exported file.
6. You can abort the execution of **Quick Export** by pressing **Abort** icon.

7. Once the export is completed, you can close the **Quick Export** window by pressing **X** icon.
Using Command-Line Export

Command-Line Export is another convenient way to export data through Data Process Logic Studio.

Usage options:

To export:

    export.exe <Data Process Logic Pack Path> <Export Template Title>

Append status messages to log file:

    export.exe <Data Process Logic Pack Path> <Export Template Title> -o <Log file>

List template titles in the work space:

    export.exe <Data Process Logic Pack Path> -l

Run all template titles in the work space:

    export.exe <Data Process Logic Pack Path> -r

Run export with parameters. Dynamically pass parameter using -p option. By default parameters are read from the xml file.

    export.exe -d <xml File Path> [-pParameterName=Value]
Below is an example of XML file's content for parameter from export template:

```xml
<root>
  <workspace path="C:\Temp\UpdateData.slfusion" />
  <template name="UpdateData">
    <mapping sourceQuery="UpdateData" destinationTable="temp">
      <queryParameters>
        <queryParameter name="MyParam1" value="40" />
        <queryParameter name="MyParam2" value="50" />
      </queryParameters>
    </mapping>
  </template>
</root>
```

Run Integration function
List Integration functions

```
export.exe <Data Process Logic Pack Path> -il
```

Run integration functions with parameters. Dynamically pass parameter using -p option. By default parameters are read from the xml file.

```
export.exe -id <xml File Path> -p [ParameterName=Value]
```

Below is an example of XML file's content for parameter from Integration Function:

```
<root>
  <workspace path="C:\Temp\UpdateData.slfusion" />
  <integrationFunctions>
    <integrationFunction name="UpdateData">
      <queryParameters>
        <queryParameter name="MyIFParam" value="30" isExpr="false" />
        <queryParameter name="MyIFParam2"
```
To use command-line export:

1. From the command prompt, go to the C:\Program Files\RISE\4.3\Fusion Studio.
2. Depending upon what you want to export, enter the command at the command prompt and press Enter.
3. The status message will be shown on the command prompt
Integration Functions Overview

Integration functions provide a way to custom tailor Data Process Logic to any data integration task. This section will describe some scenarios where you may wish to use integration functions in addition to providing more detailed specifications and examples.

Integration functions allow you to execute arbitrary code within the Data Process Logic Studio environment. Integration functions are different from user defined functions in two ways: their result can not be used in any other transformation process or export, and they consume queries and connections directly. This is useful when you wish to combine Data Process Logic data with an external process.
Integration Function API Specification

This section defines the programmatic objects that are available within integration functions.

**DataEngine Members**

**Namespace:** Data Process Logic.DataEngine

Provides runtime values of parameters within integration functions.

**Public Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDDataConnection GetConnection(string paramName)</td>
<td>Returns the connection object corresponding to the parameter paramName.</td>
</tr>
<tr>
<td>IDDataCachedTable GetQuery(string paramName, IDataParameterCollection queryParameters)</td>
<td>Returns the query object corresponding to the parameter paramName.</td>
</tr>
<tr>
<td>string GetString(string paramName)</td>
<td>Returns the string corresponding to the parameter paramName.</td>
</tr>
<tr>
<td>string Export(string templateName, IDataParameterCollection paramCollection)</td>
<td>Returns the string containing export status messages.</td>
</tr>
</tbody>
</table>
IDataConnection Members
Namespace: Data Process Logic.Connectors.ConnectorInterfaces

The data connection object must be cast to the specific connection type used. Refer to the documentation for the connector in question for the API.

IDataCachedTable Members
Namespace: Data Process Logic.Connectors.ConnectorInterfaces

Represents a collection of vectors. IDataCachedTable is commonly used to store tabular data, however it isn't required that all columns have the same length.

Public Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int GetRowCount()</td>
<td>Returns the length of the longest vector within the table.</td>
</tr>
<tr>
<td>object GetValue(int rowIndex, int colIndex)</td>
<td>Returns the value at the given row and column index.</td>
</tr>
<tr>
<td>DbTableMetadata GetMetadata()</td>
<td>Returns the table metadata, which describes the type and any display information for the fields contained within the table. Refer to the Integration Function API Specification for details on the DbTableMetadata object.</td>
</tr>
</tbody>
</table>
**DBTableMetadata Members**

*Namespace: Data Process Logic.Connectors.ConnectorInterfaces*

Represents the table metadata and display information that describes the fields within the table.

### Public Properties

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DbMetadataNamedItemTable</td>
<td>Returns table description information.</td>
</tr>
<tr>
<td>DbColumnMetadataListColumns</td>
<td>Returns the column list for this table.</td>
</tr>
<tr>
<td>DateTime LastFetchTime</td>
<td>Returns the time stamp when this metadata was last updated.</td>
</tr>
</tbody>
</table>

### Public Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int FindColumnIndex(string columnName)</td>
<td>Returns the column index from the columnName.</td>
</tr>
</tbody>
</table>
DbMetadataNamedItem Members
Namespace: Data Process Logic.Connectors.ConnectorInterfaces

Represents a named field.

Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string Name</td>
<td>Display name of this item</td>
</tr>
<tr>
<td>string Description</td>
<td>Description of this item</td>
</tr>
</tbody>
</table>

DbColumnMetadataList Members
Namespace: Data Process Logic.Connectors.ConnectorInterfaces

Collection of column metadata objects.

Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int Count</td>
<td>Returns the number of items in the list.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string[] GetColumnList()</td>
<td>Returns all the column names in the list.</td>
</tr>
<tr>
<td>Type[] GetColumnTypes()</td>
<td>Returns the types for each column in the list.</td>
</tr>
<tr>
<td>DbColumnMetadata GetColumn(int index)</td>
<td>Returns the column metadata at the specified index.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DbColumnMetadata</td>
<td>Represents the metadata for a column.</td>
</tr>
<tr>
<td>FindColumn(string colName)</td>
<td>Returns the column metadata by column name. Returns null if the column is not found.</td>
</tr>
<tr>
<td>int FindColumnOrdinal(string colName)</td>
<td>Returns the column ordinal from the column name. Returns -1 if the column is not found.</td>
</tr>
</tbody>
</table>

### DbColumnMetadata Members

**Namespace:** Data Process Logic.Connectors.ConnectorInterfaces

- **Name**
  - Gets or sets the column name.
- **ColumnOrdinal**
  - Gets the ordinal for this column.
- **Precision**
  - Gets the precision for this column.
- **Scale**
  - Gets the scale for this column.
- **ColumnSize**
  - Gets the column size in bytes for this column.
- **ColumnType**
  - Gets the datatype for this column.
- **Description**
  - Gets the description for this column.
- **PrimaryKey**
  - Returns true if this column is a primary key.
**bool** NonNullable | Returns true if this column cannot accept null values.

---

**ParamCollection Members**

*Namespace: Data Process Logic.Connectors.ConnectorInterfaces*

Represents the parameter collection.

**Public Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int Count</td>
<td>Return the number of the parameter collection.</td>
</tr>
<tr>
<td>bool IsFixedSize</td>
<td>Return true if the parameter collection is of fixed size.</td>
</tr>
<tr>
<td>bool IsReadOnly</td>
<td>Return true if the parameter collection is read-only type.</td>
</tr>
<tr>
<td>bool IsSynchronized</td>
<td>Return true if the access to the parameter collection is synchronized (thread-safe).</td>
</tr>
<tr>
<td>object SyncRoot</td>
<td>Get the object that can be used to synchronize access to parameter collection.</td>
</tr>
<tr>
<td>Parameter ParamCollection[int index]</td>
<td>Get the parameter in the parameter collection at the specified index.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Get the parameter in the parameter collection with the specified name.</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>ParamCollection[string parameterName]</td>
<td></td>
</tr>
</tbody>
</table>

**Public Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int Add(object value)</td>
<td>Add parameter.</td>
</tr>
<tr>
<td>void Clear()</td>
<td>Clear all parameters in the collection.</td>
</tr>
<tr>
<td>bool Contains(string parameterName)</td>
<td>Check if the parameter collection contains the specified parameter's name.</td>
</tr>
<tr>
<td>void CopyTo(Array array, int index)</td>
<td>Copy the range of array to the parameter collection starting at the specified index.</td>
</tr>
<tr>
<td>System.Collections.IEnumerator GetEnumerator()</td>
<td>Return an enumerator for a section of a parameter collection.</td>
</tr>
<tr>
<td>int IndexOf(string parameterName)</td>
<td>Return the index of the parameter in the parameter collection.</td>
</tr>
<tr>
<td>void Insert(int index, object value)</td>
<td>Insert a parameter into the parameter collection at the specified index.</td>
</tr>
<tr>
<td>Parameter Item[int index]</td>
<td>Return parameter of the specified index.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>void Remove(object value)</td>
<td>Remove the first occurrence of a specified object from the parameter collection.</td>
</tr>
<tr>
<td>void RemoveAt(string parameterName)</td>
<td>Remove the specified parameter.</td>
</tr>
<tr>
<td>int TypeSafeAdd(IDataParameter prm)</td>
<td>Add parameter to the end of a parameter collection.</td>
</tr>
<tr>
<td>void TypeSafeInsert(int index, IDataParameter prm)</td>
<td>Insert a parameter into a parameter collection at the specified index.</td>
</tr>
</tbody>
</table>
Using the Integration Function Editor

The integration function editor allows you to develop and test custom integration logic within the **Data Process Logic Studio** environment. The integration function editor is identical to the user function editor with the addition of two features: parameters and the ability to execute code. For details on how to write integration functions, refer to the examples and API specifications.
Parameters are accessed by using the **Function parameters** tool in the toolbar. This will display the following dialog. Parameters can be of type **Connection**, **Query**, or **String**.

The parameter name is the name used in code to reference this parameter, and must be unique. You can add and remove parameters by using the **Add** and **Remove** buttons respectively.

To check the syntax of your code, use the **Check syntax (F4)** tool in the tool bar. To run your integration function code, use the **Run Code (F5)** tool in the toolbar. Be aware that there is no protection against your code performing destructive operations. To stop execution of code, you can use the **Stop execution** tool at any time. Also note that **System.Console** is routed to the output.
pane, and any text that is printed to the console will appear in that pane.
Integration Function Examples

This section provides a sample integration function for reference.

Example1: Fetching a parameter value and exporting data.
This is a simple example to demonstrate the use of the integration function features.
This example has a query which takes a string parameter. So we have created two function parameters "MYIFParam" of type string and "MyIFTable" of query type. Then we pass the string parameter to the query. We also have an export template called "ExportTemplate" which exports the queries result to a text file. The export status messages are displayed in the output.

[c#]
using System;
using Data Process Logic.DataEngine;
using Data Process Logic.Connectors.ConnectorInterfaces;
public class DemoParameters : IIntegrationFunction {
    private class StatusHandler : IDataQueryStatus {
        public int Push(string source, string status, string msg) {
            return 0;
        }
        public void Pop(int pushId) {
        }
    }
}
public void Status(string source, string status, string msg,
StatusType statusType) {
}

public bool IsOperationCancelled() {
    return false;
}

public void ChangeStatus(int pushId, string status, string msg)
{
}

public int BeginRowOperation(string source, string status,
string msg, long maxRowHint) {
    return 0;
}

public void CurrentRow(int rowOpId, long rowNo) {
}

public void EndRowOperation(int rowOpId, long rowsProcessed) {
}

public void Execute(DataEngine theEngine) {
    string userParam =
    theEngine.GetString("MyIFParam");
Console.WriteLine(userParam);
ParamCollection coll = new ParamCollection();
coll["MyParam"] = int.Parse(userParam);
IDataCachedTable tbl =
theEngine.GetQuery("MyIFTable", coll);

Console.WriteLine(tbl.GetValue(2, 0));

string msgStack = theEngine.Export("ExportTemplate",
coll, new StatusHandler());

Console.WriteLine(msgStack);

string userParam1 =
theEngine.GetString("MyIFParam2");

Console.WriteLine(userParam1);

coll = new ParamCollection();
coll["MyParam"] = int.Parse(userParam1);

tbl = theEngine.GetQuery("MyIFTable", coll);

Console.WriteLine(tbl.GetValue(2, 0));

string userParam2 =
theEngine.GetString("MyIFDateParam");

Console.WriteLine(userParam2);

}
Export Status Messages:

Query Data Manager: Created local store
C:\DOCUME~1\sheetal\LOCALS~1\Temp\Fusion2.db

Data Export: Fetching data from query: MyTable
Data Export: MyTable --> temp

Text Connection: Export Started
Data is being exported

Text Connection: Export Completed
Exported 3 rows in 0h 0m 0s

Data Export: Update successfully completed

Query Data Manager: Removed local store
C:\DOCUME~1\sheetal\LOCALS~1\Temp\Fusion2.db
User Functions Overview

User functions allow you to extend the transformation function library with custom functions. User functions are defined and stored within a Data Process Logic package and can be authored and tested all within the Data Process Logic environment.

There are 3 general types of parameters a user function can accept and return: IVector, IData CachedTable and Scalar. Scalars are used directly as value types such as int, double, string etc. The IVector and IData CachedTable interfaces are defined in the API section below. Both of these interfaces are defined in the Data Process Logic.Connectors.ConnectorInterfaces namespace. If you are using these types, you need to include the namespace in your code.
User Function API Specification

This section defines the programmatic objects that are available within user functions.

IVector Members

Namespace: Data Process Logic.Connectors.ConnectorInterfaces

Represents a collection of scalar values, usually as a column within a table.

Public Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public string GetName()</td>
<td>Returns the display name for this vector.</td>
</tr>
<tr>
<td>public voidSetName(string name)</td>
<td>Sets the display name for this vector.</td>
</tr>
<tr>
<td>public TypeGetDataType()</td>
<td>Returns the data type for the values contained in the vector.</td>
</tr>
</tbody>
</table>
IDataCachedTable Members

Namespace: Data Process Logic.Connectors.ConnectorInterfaces

Represents a collection of vectors. IDataCachedTable is commonly used to store tabular data, however it isn't required that all columns have the same length.

Public Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public int GetRowCount()</td>
<td>Returns the length of the longest vector within the table.</td>
</tr>
<tr>
<td>public object GetValue(int rowIndex, int colIndex)</td>
<td>Returns the value at the given row and column index.</td>
</tr>
<tr>
<td>DbTableMetadata GetMetadata()</td>
<td>Returns the table metadata, which describes the type and any display information for the fields contained within the table. Refer to the Integration Function API Specification for details on the DbTableMetadata object.</td>
</tr>
</tbody>
</table>
Using the User Function Editor

The user function editor allows you to author and syntax check your custom code. To execute a user function, you must create a transformation query that uses your function. The editor contains two panes: the **Code editor**, and the **Output** pane.

Details of how to write user functions are available in the examples and API specifications within this section. To check the syntax of your function, use the **Check syntax (F4)** tool within the tool bar. The results of the syntax check will appear in the **Output** pane. Double clicking on a warning or error in the **Output** pane will position the cursor to the correct line and character position.
User Function Examples

This section provides a sample user function for reference.

Example 1: Computing the sum of a vector

This example will sum the non-null values of a vector and return the results as a scalar. There are a few nuances worth pointing out with this example. First, is that all values are cast to integer. If you wish to sum a vector of floating point values, this function will create significant precision loss. To accommodate input values of different types, check the type returned by IVector.GetValue before doing your processing. The second item worth noting is error handling. Data Process Logic Studio is tolerant of user functions throwing exceptions, however, no data will be returned if an exception is thrown. For this reason, be sure that your user functions are as robust as possible.
using System;
using Data Process Logic.Connectors.ConnectorInterfaces;

public class Sum {
    public int Execute(IVector v) {
        //int rowCount = v.GetRowCount();
        ITableIterator iter = v.GetIterator();
        int sum = 0;
        for (int i=0; iter.Read(); i++) {
            object currentVal = v.GetValue(iter);
            if (System.DBNull.Value == currentVal) {
                continue;
            }
            sum += Convert.ToInt32(currentVal);
        }
        return sum;
    }
}
Imports System
Imports Data Process Logic.Connectors.ConnectorInterfaces

Public Class ExampleSum
    Public Function Execute(v As IVector) As Int32
        Dim iter As ITableIterator = v.GetIterator()
        Dim sum As Int32 = 0
        While iter.Read()
            Dim currentVal As object = v.GetValue(iter)
            If Not currentVal.Equals(System.DBNull.Value) Then
                sum += Convert.ToInt32(currentVal)
            End If
        End While
        Return sum
    End Function
End Class
public class ExampleSum {
    public function Execute(v : IVector): long {
        var rowCount : int = v.GetRowCount();
        var sum : long = 0;
        for (var i=0; i<rowCount; i++) {
            var currentVal : Object = v.GetValue(i);
            if (System.DBNull.Value == currentVal) {
                continue;
            }
            sum += Convert.ToInt64(currentVal);
        }
        return sum;
    }
}
Using Data Process Logic as a data bridge

Data Process Logic Studio can function as a powerful data bridge to move data between otherwise incompatible applications. Data Process Logic Studio allows you to select and manipulate the data from the original data source in ways that may not be available within reporting packages. Once the data has been defined, it can be consumed by other applications without opening Data Process Logic Studio.

When used in this manner, Data Process Logic:

- Consumes the data through its own internal data connectors.
- Filters, combines, and manipulates the data through queries.
- Makes the data accessible in its role as an OLE DB provider.
Configuring Data Process Logic as an OLE DB data source

The installation process automatically registers Data Process Logic as an OLE DB provider, allowing you to use an Data Process Logic package as a data source in any software application compatible with OLE DB data sources. However, you must set up the individual Data Process Logic packages as data sources within the consuming application.

Most OLE DB consumers use the Windows Data Link dialog or something similar to set up data sources. To specify an Data Process Logic package as a data source, you need only enter the full path to the Data Process Logic package as the Data Source. You need not enter anything in the Location field. Use the Test Connection button, if available, to verify that the connection is working.
Example: Using Data Process Logic as a data bridge with Microsoft® Excel

Microsoft® Excel is the everyday workhorse for many businesses; however, it is limited in its import capabilities. By using Data Process Logic Studio as a data bridge between Excel and other applications, you can expand Excel’s reach considerably.

The procedure below provides general descriptions for using Data Process Logic Studio with Excel. The exact steps may vary with differing versions of Excel.

**Data Process Logic Studio** as a data bridge To use to Excel

1. To configure Data Process Logic package as your data source, go to the Data menu in Microsoft Excel sheet and select Import External Data >Import Data.

2. In the Select Data Source dialog that appears, select +Connect to new Data Source.odb and click Open.

3. In the Data Connection Wizard, select Other/advanced as the kind of data source you want to connect to, and then click Next.

4. On the Provider tab of the Data Link Properties dialog that appears, select RISE Data Process Logic Data Bridge and click Next.
5. On the **Connection** tab, type the full path to the desired **Data Process Logic** file or use the **Browse** button to locate and select the file. Use the **Test Connection** button to verify the connection, then click **OK**.

6. A new dialog appears, showing as tables all the queries available from the **Data Process Logic** package.

![Data Connection Wizard](image)

7. Select the desired queries and click **Next**.

8. When you return to the main Excel screen, the data from the **Data Process Logic** connection appears on the current worksheet.
Data Process Logic Studio supplies a specific expression syntax for building transform queries (Standard queries also use the Data Process Logic expression syntax from within the Expression Editor).

When using the Data Process Logic Studio syntax, you will combine data types, representing various forms of information, within a specific grammar. Data Process Logic Studio expression syntax employs a grammar composed of literals, operators, functions, tables, and columns. The grammar describes the rules that govern how expressions must be formatted and how data types interact with each other.

You can experiment with the expression syntax by creating transform queries within Data Process Logic’s transform editor.
and previewing the results in the output pane at the bottom of the editor.
Expression syntax: data types

The Data Process Logic Studio expression system uses three different data types:

- **Scalar (one-dimensional):** single values, including numeric, date/time, and string values
- **Vector (two-dimensional):** a list of scalar (individual) values
- **Table (two-dimensional):** a list of vector values

Most Data Process Logic functions and operators specify particular data types, but are tolerant of using multiple types. If you provide a type other than the one specified for a given function or operator, Data Process Logic Studio attempts to process the type correctly, using the type conversion rules described below.

Type conversion rules

When you provide a value with an unexpected data type, Data Process Logic Studio sometimes must convert from one type to another in order to process the expression. Specific rules govern the conversion when using functions, transform queries, and if-then-else statements/operators.

**Note:** when converting string data type to int, date, etc. using our transformation functions, if the input string is not in the right format, we may get an error “input string is not in right format.”
Functions

Most functions expect a scalar but will also accept a vector. If vector data is provided, the function evaluates each item in the source vector and return a new vector as the result.

Transform queries

Data Process Logic Studio always converts the results of a transform query to a table, regardless of the data type returned by the expression.

If-then-else statement and operators

If-then-else statements and operators can take either scalar or vector type data. If vector data composes one side of an operator and scalar data composes the other side, Data Process Logic applies the scalar value to all items in the vector. If both sides contain vectors, Data Process Logic Studio evaluates equal row indices against one another. An if-then-else statement returns the appropriate then-or-else values in a vector the same length as the source vector.
Expression syntax: grammar components

Literals
A literal is the text representation of a scalar (singular) value, such as the number 5 or the string “text”. Note that string literals must be within double-quotes to be valid. A literal by itself is a valid expression, although not a particularly interesting one.

Tables
Since queries present their data in tabular form, functions that require table parameters can take queries as input. You can represent a query within an expression by using double square-brackets around the query name.
Example:

```plaintext
[[SampleQuery]] (table contents of SampleQuery)
```

Columns
**Data Process Logic Studio** expression syntax represents columns as vectors by enclosing the query and column name in single square brackets. A column name must be preceded by the name of the query it belongs to using the dot “.” character. The query and column names can be placed within the same pair of brackets or in separate bracket pairs. You can use a column in this manner exactly as you would any vector data.
<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[SampleQuery.SampleColumn]</code></td>
<td>(contents of SampleColumn)</td>
</tr>
<tr>
<td><code>[SampleQuery].[SampleColumn]</code></td>
<td>(contents of SampleColumn)</td>
</tr>
</tbody>
</table>
Expression syntax: operators

Data Process Logic uses three types of operators in its expression syntax: arithmetic, relational, and logical.

Arithmetic operators

Arithmetic operators are represented by the symbols + (addition), - (subtraction), / (division), * (multiplication), and % (modulus—the remainder after dividing two integers). The operators can be applied to scalar and vector data.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>2 + 3</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>2 - 3</td>
<td>-1</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>4/2</td>
<td>2</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>2 * 3</td>
<td>6</td>
</tr>
<tr>
<td>%</td>
<td>Modulus</td>
<td>5 % 3</td>
<td>2</td>
</tr>
</tbody>
</table>
Relational operators
Relational operators compare two values and return the result of the comparison as a Boolean value of True or False. Data Process Logic supports the six operators listed below. Data Process Logic supports the common database operators In and Between as functions;

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>equals</td>
<td>0 = 1</td>
<td>False</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>not equals</td>
<td>0 &lt;&gt; 1</td>
<td>True</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>0 &gt; 1</td>
<td>False</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>0 &lt; 1</td>
<td>True</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>0 &gt;= 1</td>
<td>False</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>0 &lt;= 1</td>
<td>True</td>
</tr>
</tbody>
</table>

Logical operators
Logical operators (also called propositional operators) operate against two Boolean values. Data Process Logic supports the operators And, Or, and Not, as described below.

Data Process Logic Studio User Guide 199
<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>both statements are true</td>
<td>(5&gt;3) And (5&gt;4)</td>
<td>True</td>
</tr>
<tr>
<td>Or</td>
<td>at least one statement is true</td>
<td>(5&lt;3) Or (5&gt;4)</td>
<td>True</td>
</tr>
<tr>
<td>Not</td>
<td>neither statement is true</td>
<td>Not ((5&lt;3) Or (5&gt;4))</td>
<td>False</td>
</tr>
</tbody>
</table>
Expression syntax: functions

Functions provide a rich tool set for creating useful expressions. [Data Process Logic Studio] includes an ever-growing number of built-in functions in various categories. Details on specific functions are available in [Data Process Logic Function Reference Guide].

The syntax for any function takes the form:

FunctionName(param1, param2, ...)

Most [Data Process Logic] functions are very tolerant in accepting a range of data types. For example, the function [Upper] converts a string to uppercase. If given a scalar string, will [Upper] returns a scalar result (e.g., cost becomes COST). If given a vector of strings, [Upper] returns a vector with all strings converted to uppercase (e.g., price, units, cost becomes PRICE, UNITS, COST).

The arithmetic operators work in a similar way. If you multiply a vector by a Scalar, the result will be a vector with each item multiplied by the scalar. If you multiply a vector by a vector, [Data Process Logic] returns a vector where matching row indices have been multiplied by one another.
<table>
<thead>
<tr>
<th>Function</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>Abs(-5)</td>
<td>5</td>
</tr>
<tr>
<td>Sum</td>
<td>Sum(Array(1,2,3))</td>
<td>6</td>
</tr>
<tr>
<td>Upper</td>
<td>Upper(&quot;a&quot;)</td>
<td>A</td>
</tr>
<tr>
<td>Upper</td>
<td>Upper(Array(&quot;a&quot;, &quot;b&quot;, &quot;c&quot;))</td>
<td>A B C</td>
</tr>
</tbody>
</table>
Expression syntax: other language features

The Data Process Logic Studio expression system supports the use of arrays and if-then-else statements to create expressions not otherwise available. It also provides the Fusion TestTable query to create sample data for testing.

Arrays

An array takes a list of scalar data values and produces a corresponding vector. This feature allows you to create vector types manually.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array(“a”, “b”, “c”)</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>c</td>
</tr>
<tr>
<td>Array(1,2,3)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

If-Then-Else

You can apply simple logic to expressions using if-then-else. The syntax takes the form:

\[
\text{If(condition, value if true, value if false)}
\]
Data Process LogicTestTable

In order to provide sample data for testing, you can use Data Process LogicTestTable like a query (e.g. [[FusionTestTable]]) to create a table of sample data that includes all data types (integer, string, etc.)
RISE makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

RISE shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

This document is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without prior written consent of RISE. The information contained in this document is subject to change without notice.

This publication contains references to the products of SAP AG. SAP R/3, SAP NetWeaver, SAP Business One, ByDesign, SAP BusinessObjects Explorer, and other SAP products and services.
and/or registered service marks of Intuit Inc., or one of its subsidiaries, in the United States and other countries.

Sage, Sage ACT! among others, are registered trademarks and/or registered service marks of Sage or one of its subsidiaries, in the United States and other countries.

RISE Incorporated
2925 Skyway Circle North
Irving, Texas 75038