Varigence Documentation

Cover Page sub header

Table of Contents

- BimlStudio Documentation
- BimlStudio User Guide
  - AdventureWorks LT Walkthrough
    - Setting up for the Sample
    - Creating a New Project
    - Creating a New Connection
    - Saving a Project
    - Creating a New Database
    - Importing from the Sample Database
    - Creating a New Schema
    - Creating a Basic Table
    - Creating a Dimension Table
    - Adding Existing Files to a Project
    - Creating a Fact Table
    - Deploying Tables to SQL
    - Preparing to Build the Example Project
    - Building the Example Project
  - Editors Overview
    - Connection
    - Package
    - Table
    - Static Source
    - Schema
    - Principal
    - File Format
    - Cube
    - Action
    - KPI
    - Perspective
    - Aggregation Design
    - Partition
    - Calculation
    - Script Project

BimlScript
Biml
Common Tool Windows
Package Tool Windows
Miscellaneous
Importing Tables using BimlScript
Creating A Basic Package
Switching Build Types for Biml Files
Configuring Project Settings
Biml Compiler Command Line Options
Source Control Setup
Create an Expandable Transformer
Choose The Right Script Mode
Using Configuration Files
Project View Idiosyncrasies
BimlStudio User Guide
Release Notes
BimlStudio 2018.1
BimlStudio 2017
Mist 4.0 Update 1
Mist 4.0
Mist 3.4
Mist 3.3
Mist 3.2
Mist 3.1
Mist 3.0
Mist 2.0

BimlStudio Documentations

BimlStudio User Guide

The BimlStudio User Guide describes all aspects of the BimlStudio user interface.

The BimlStudio User Guide is broken up into three parts. The first part provides a step by step walkthrough for building assets for use in a sample cube, leveraging the AdventureWorksLT database. The second part describes each BimlStudio editor, tool window, and ribbon button. The third part includes miscellaneous topics that don't fit into the first two groups.

BimlStudio Forum

If you have questions that aren't addressed in this guide, you can search and post in the BimlStudio forum for help.

Email Support

While the forums are the best way to get answers, if your question contains proprietary, or other, information that you would prefer not to post publicly, private email support is available at support@varigence.com

BimlStudio User Guide

The BimlStudio User Guide describes all aspects of the BimlStudio user interface.

The BimlStudio User Guide is broken up into three parts. The first part provides a step by step walkthrough for building assets for use in a sample cube, leveraging the AdventureWorksLT database. The second part describes each BimlStudio editor, tool window, and ribbon button. The third part includes miscellaneous topics that don't fit into the first two groups.

BimlStudio Forum

If you have questions that aren't addressed in this guide, you can search and post in the BimlStudio forum for help.

Email Support

While the forums are the best way to get answers, if your question contains proprietary, or other, information that you would prefer not to post publicly, private email support is available at support@varigence.com

AdventureWorks LT Walkthrough

This section of the user guide walks through common operations in Mist, with detailed instructions and screenshots to explain how to accomplish those tasks.

Setting up for the Sample

To build the sample in this walkthrough, some initial setup is required.

1. You will need to install the AdventureWorks LT database on your SQL server. This sample database is available on CodePlex. Because the database sample is dependent on your version of SQL Server, you will need to download the appropriate version.

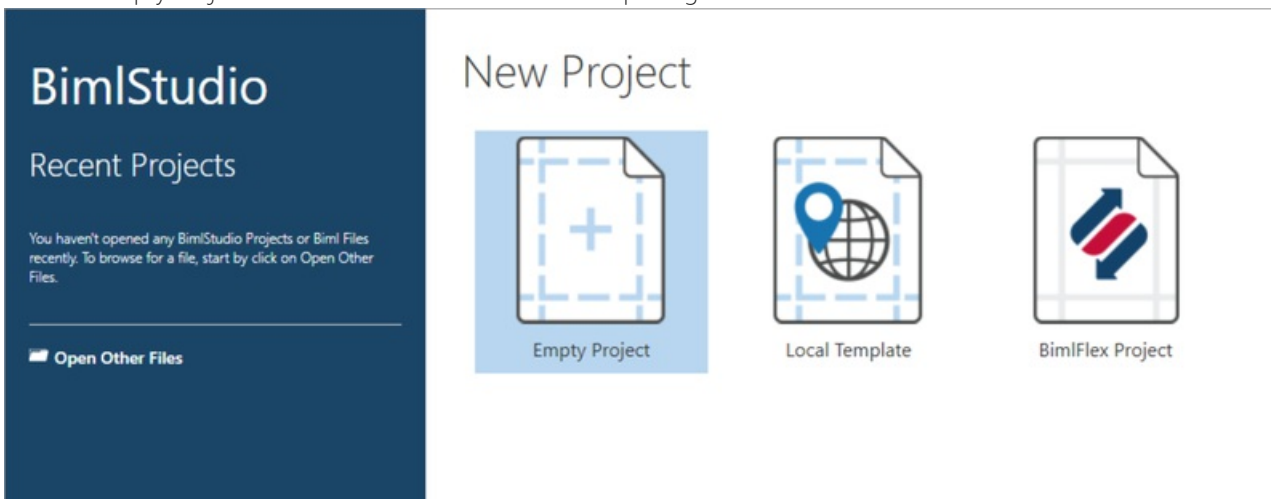
2. Download samples databases from here: https://msftdbprodsamples.codeplex.com/

3. You will also need to create a database on your SQL server named AdventureWorksLTDataMart.

A BimlStudio project lets you define all the assets you need to build your BI solution.

To create a new project in BimlStudio:

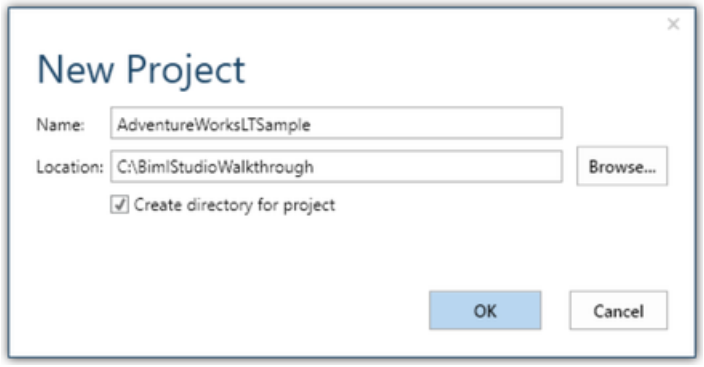
1. Click the Empty Project button on the home screen after opening BimlStudio.



2. The New Project dialog will open. Click on the Browse button.

3. This opens the Project Location dialog. Use it to select the folder where the BimlStudio project should be created and then click Select Folder.

4. BimlStudio inserts the selected folder path in the Location textbox. Enter the project name in the Name textbox and press OK.



This dialog creates a folder, with the same name as your project, under the Location path. The folder contains the project file and subdirectories for holding Biml files and compiled output.

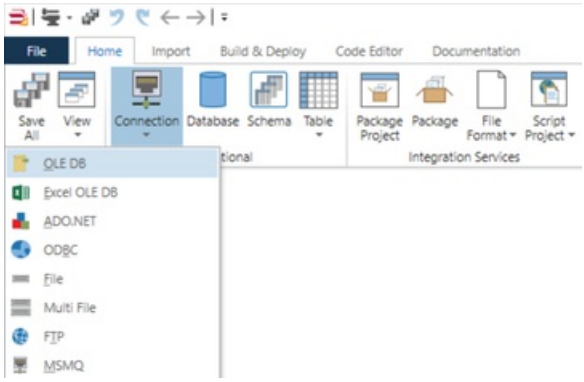
You have now created a project and are ready to add assets.

Creating a New Connection

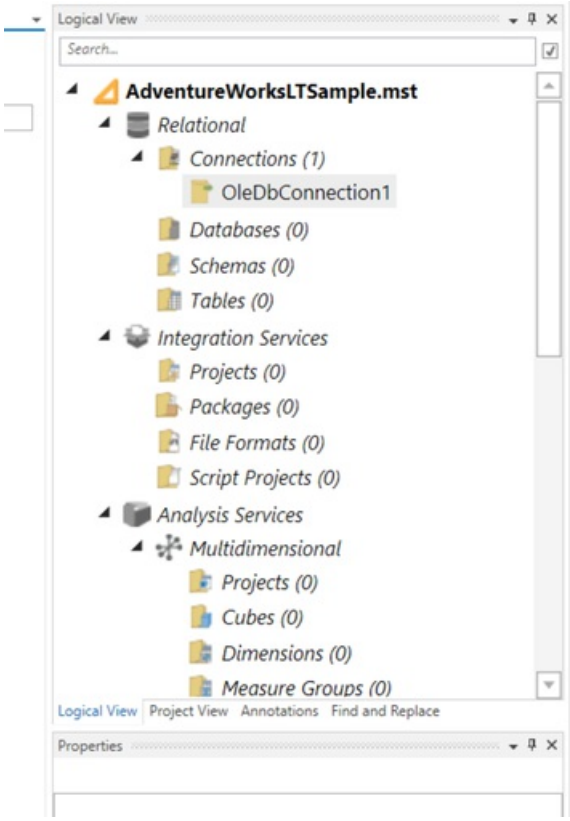
In BimlStudio, connections are defined once in a project and shared with all the objects that need to reference them. This allows you to easily update connections in a single location.

Create a new connection

1. Go to the Home tab on the ribbon, and click the **Connection** button. The Connection button is a split button. Clicking the top half will create a new OLE DB connection, and clicking the bottom half will list all the available connection types.



2. Click the **OLE DB** menu item to create an OLE DB Connection. When the connection is created, it will appear in the Logical View under Connections.



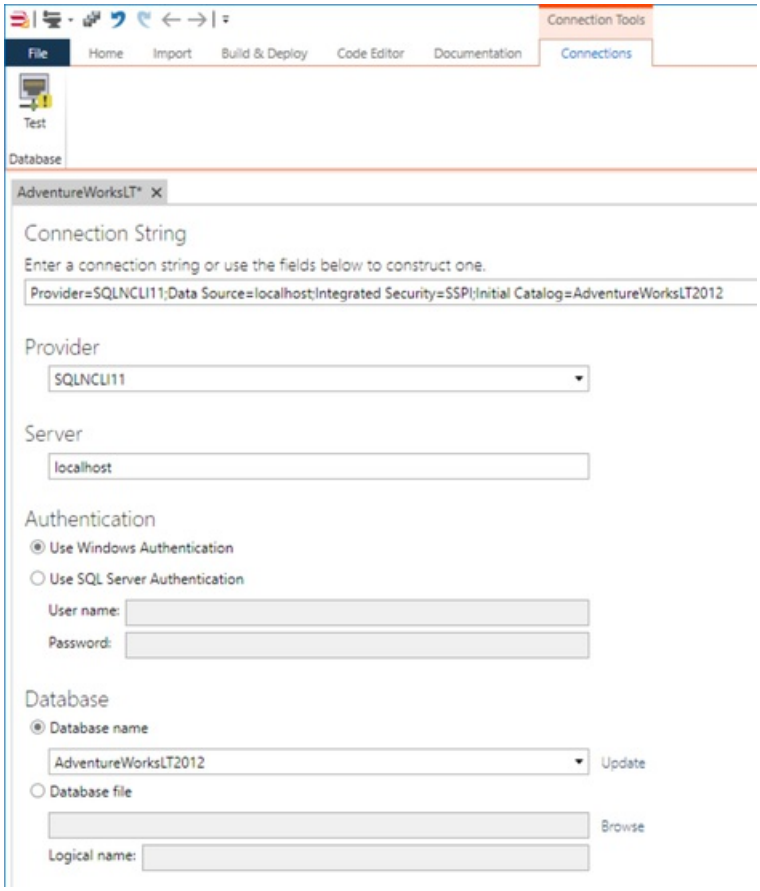
3. The connection designer should be open on the right, if not Double-click on the connection to open the designer for it.

4. OleDbConnections require that you provide a connection string. For this example, you can copy, paste and edit the following connection string:

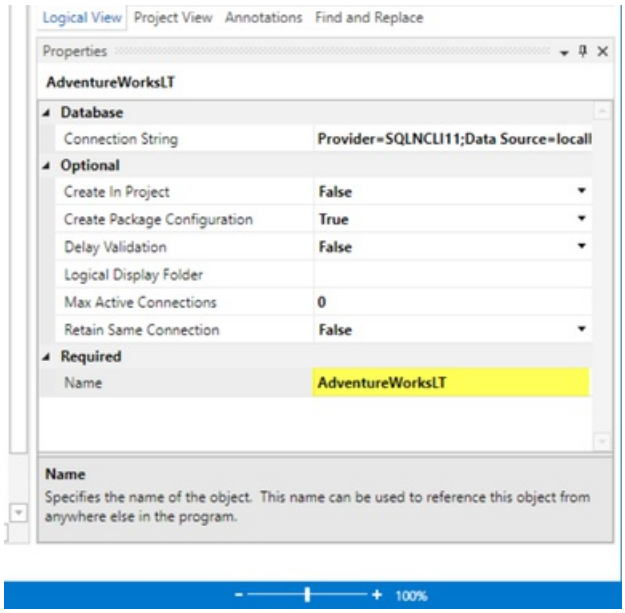
```
Provider=SQLNCLI11;Data Source=localhost;Integrated Security=SSPI;Initial Catalog=AdventureWorksLT2012;
```

5. You can use the connection builder by following these steps:

- Select a **Provider**
- Specify a **Server**
- Choose and **Authentication** method
- Specify a **Database name** or click the **Update** link to get a list of available databases



6. Change the Name value from OleDbConnection1 to AdventureWorksLT to give the connection a meaningful name. You will be prompted "Do you also want to rename the asset's Biml file?" click Yes



7. Save the project to persist your changes to the project files. See Saving a Project for more information.

8. Once you've entered a connection string, you can click the Test button in the Connection Tools. Connections tab in order to validate it.

The connection string in the example above allows you to connect to the AdventureWorksLT database running on your local

computer, using your Windows credentials. You will need to alter the connection string if you are using another server or a different authentication method.

http://connectionstrings.com is a good website to determine what type of connection string you should use for various database systems.

View Biml

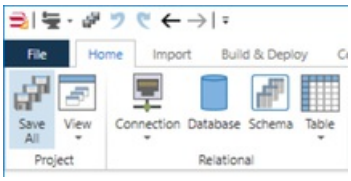
If you right click the connection and click **View Biml** it should look similar to the following snippet.

```
    <Biml xmlns="http://schemas.varigence.com/biml.xsd">
        <Connections>
            <OleDbConnection Name="AdventureWorksLT" ConnectionString="Provider=SQLNCLI11;Data
Source=localhost;Integrated Security=SSPI;Initial Catalog=AdventureWorksLT2012" />
        </Connections>
    </Biml>
```

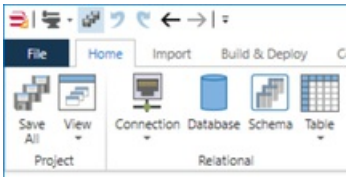
Saving a Project

There are several options for saving a project in BimlStudio.

- Press the CTRL+S key combination.
- Click the Save button on the Home tab of the ribbon.



- Click the Save button on the Quick Access Toolbar, which is located at the far upper left of the application window. The Quick Access toolbar is visible regardless of which ribbon is currently shown.

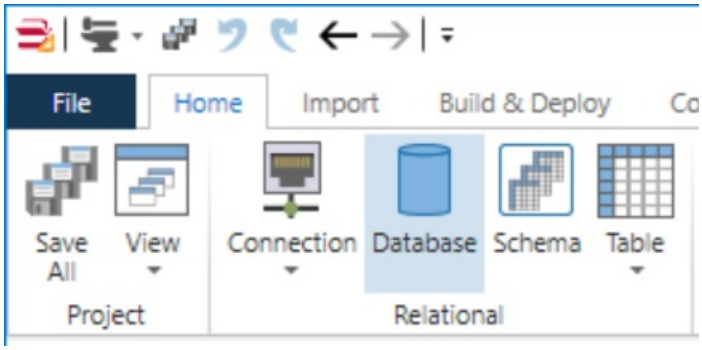


Creating a New Database

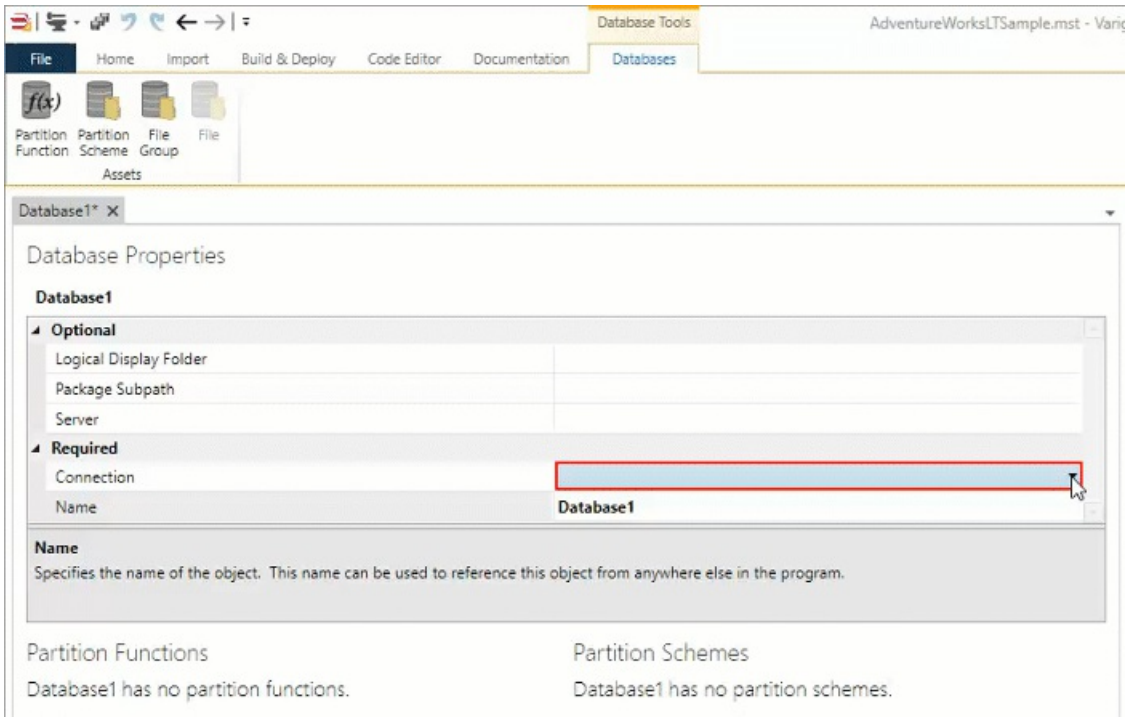
Databases contain relational objects, such as tables, schemas, and views. Any relational object in a database automatically uses the database's connection.

Create a new database

1. Go to the Home tab on the ribbon, and click the Database button.



2. The Database Properties should be open on the right, if not Double-click on the database to open the designer for it.
3. Databases require that you specify a connection. For this example, use the AdventureWorksLT connection.
4. Change the Name value from Database1 to AdventureWorksLTDatabase to give the database a meaningful name.



5. Save the project to persist your changes to the project files. See Saving a Project for more information.

View Biml

If you right click the connection and click **View Biml** it should look similar to the following snippet.

```
<Biml xmlns="http://schemas.varigence.com/biml.xsd">
    <Databases>
        <Database Name="AdventureWorksLTDatabase" ConnectionName="AdventureWorksLT" />
    </Databases>
</Biml>
```

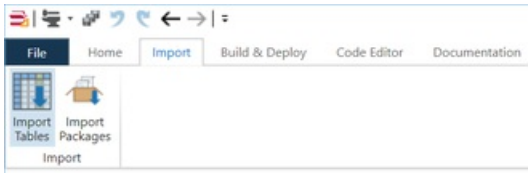
Importing from the Sample Database

BimlStudio provides an import capability. This allows you to pull existing schema and table definitions from a database into your model. These items are then imported as new objects in the model. Once imported, you can edit them as needed.

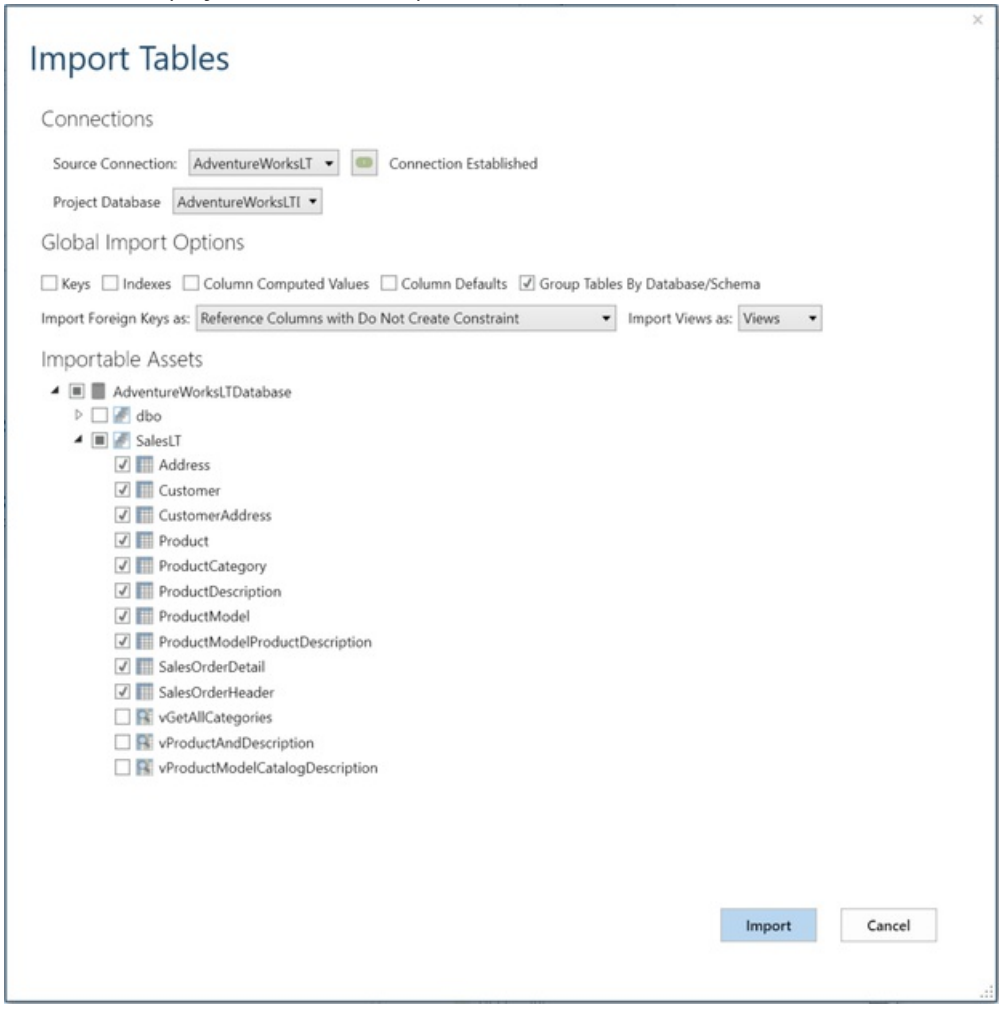
To import database assets, you first need to create a connection to the database. See Creating a Connection for more information. Then, you then need to create a database asset. See Creating a Database for more information.

Import Tables using BimlStudio Ribbon

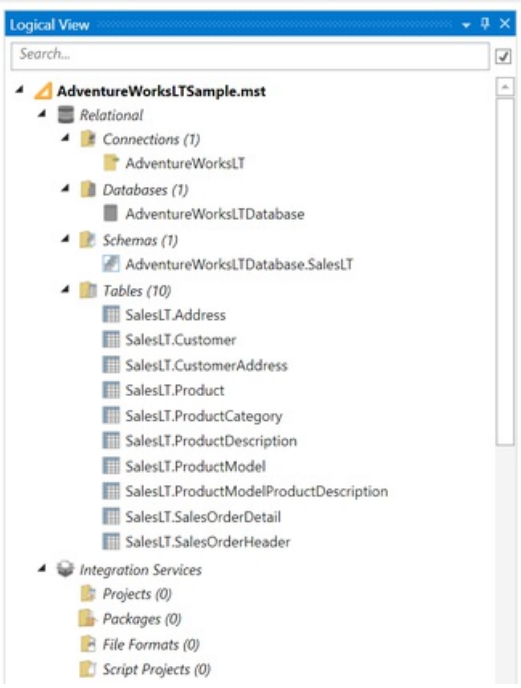
1. Select the Import tab on the ribbon.
2. Click the Import Tables button.



3. Select the connection to use for importing (AdventureWorksLT in this example). The connection state indicator will turn green to indicate a good connection.
4. Also select the project database to import the schemas and tables into (AdventureWorksLTDatabase in this example).



5. Select the schemas and tables that you would like to import. For this example, all tables (not views) for SalesLT schema should be selected.
6. Click **Import** to begin the import.
7. Once the import is complete, there will be 10 tables under the Tables folder in the logical view, as well as 1 schema under

the Schema folder. 

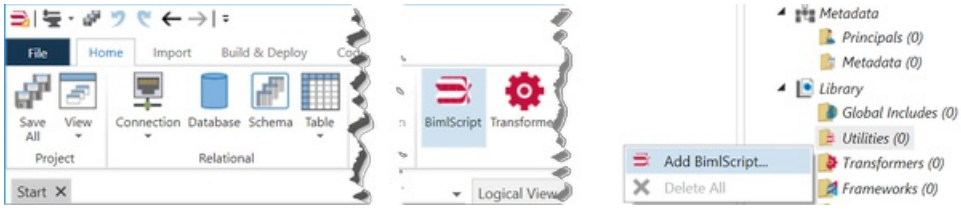
8. When it completes, save the project to persist your changes.

Import Tables using BimlScript

BimlStudio allows you to import schema and table definitions from a database, along with updating your model with the current state of the definitions at each compile. To leverage the update capability, you will need to use BimlScript to author the import logic; the BimlScript is executed during each compile, obtaining the current state of schemas and tables.

You can import database assets by following these steps:

1. Select the Home tab in the ribbon and click the BimlScript button. You can also right click the Utilities folder and click **Add BimlScript**. This will open the **New Item** dialog, click **Add**.



2. This creates a BimlFile that's added in the Logical View under Library\Utilities.

3. The BimlFle should be open on the right, if not Double click on the BimlFile to open the BimlScript designer.

4. Enter your script in the **BimlScript Input Editor** pane. When executed, the script generates Biml for your imported assets. Clicking on the notification bar saves the script and displays the Biml, resulting from running the script, in the **Preview Expanded BimlScript** pane.

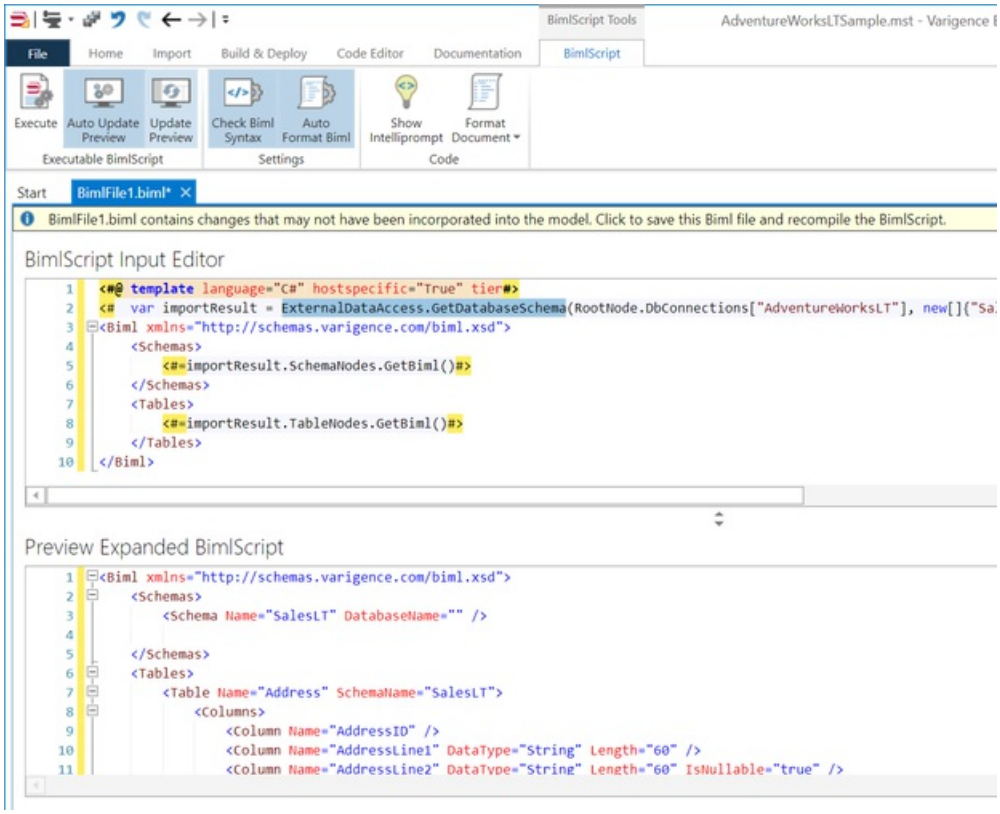
5. For importing assets from the AdventureWorksLT database, you can copy and paste this BimlScript into the **BimlScript Input Editor** pane. You can also use this sample as a starting point for writing your own import script.

```
        <#@ template language="C#" hostspecific="True" tier#>
        <# var importResult =
ExternalDataAccess.GetDatabaseSchema(RootNode.DbConnections["AdventureWorksLT"], new[]{"SalesLT"}, null,
ImportOptions.ExcludeForeignKey | ImportOptions.ExcludeColumnDefault | ImportOptions.ExcludeViews |
ImportOptions.ExcludeIdentity);#>

        <Biml xmlns="http://schemas.varigence.com/biml.xsd">
            <Schemas>
                <#=importResult.SchemaNodes.GetBiml()#>
            </Schemas>
            <Tables>
                <#=importResult.TableNodes.GetBiml()#>
            </Tables>
        </Biml>
```

```
1.  The first thing to notice is that all code fragments are surrounded by required <# and #> characters.
2.  The first line is the required template directive. The language parameter depends on the language you're
using; this example uses C#.
3.  The importResult variable returns a collection of type ImportResults. The property's getter uses RootNode,
which is the root of the project tree; all assets in your project can be reached from RootNode. The
Connections collection returns the _AdventureWorksLT_ connection and used by
ExternalDataAccess.GetDatabaseSchema call method to perform the import.
5.  Within the Biml element, the Schemas and Tables lists are populated by retrieving the SchemaNodes and
TableNodes collections from the Results property and calling GetBiml, which converts the returned schema and
table items into Biml.
6.  To learn more about these functions, check out the Biml API documentation [here.]
(http://www.varigence.com/documentation/biml/api.html)
```

1. While writing the script, you can click on the notification bar to save the BimlScript and examine its output. The Update Preview ribbon button also refreshes the Output editor.



2. Once the output looks correct, press the Execute button in the ribbon to import the assets into your model.

3.

Creating a New Schema

Schemas group table assets.

Create a new schema

1. Begin by creating a new Connection. To review the steps for creating a connection, see the Creating a Connection topic. For this example, create a connection named AdventureWorksLTDataMart using the following connection string:

   ```
   Provider=SQLNCLI11;Data Source=localhost;Initial Catalog=AdventureWorksLTDataMart;Integrated Security=SSPI;
   ```

   This connection points to a database that will be created later in the example project.

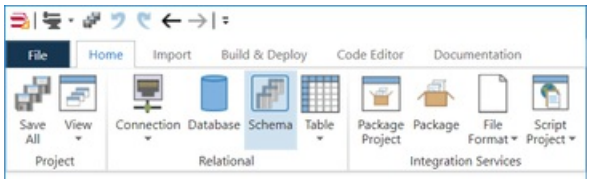
2. Next, create a new Database. To review the steps for creating a database, see the Creating a Database topic. For this example, name the database:

   `AdventureWorksLTDataMartDatabase`

   Assign its connection property to

   `AdventureWorksLTDataMart`

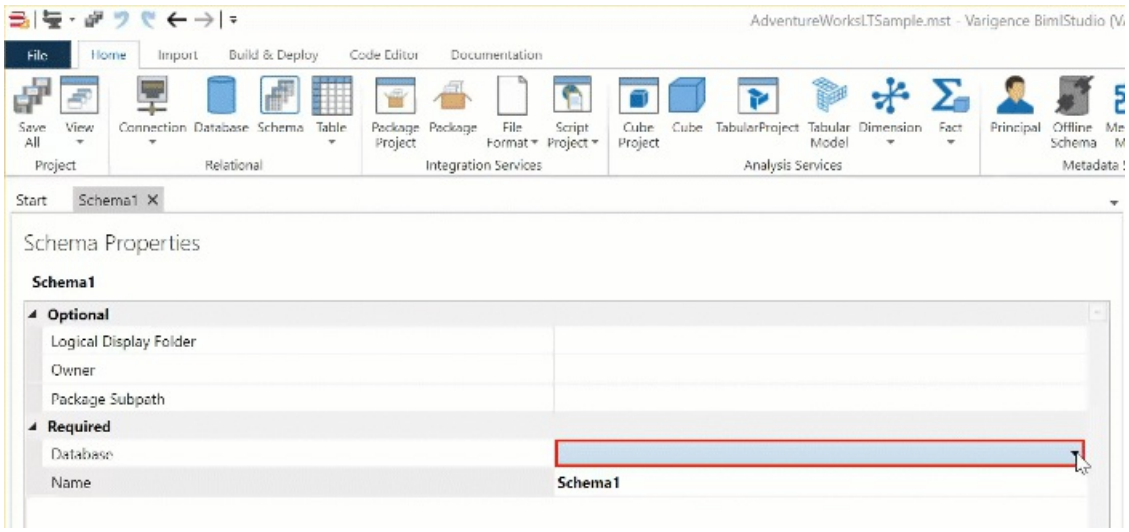
3. Go to the Home tab on the ribbon, and click the Schema button.



4. The Schema Properties should be open on the left, if not Double-click on the schema to open the designer for it.

5. Schemas require that you specify a database. For this example, use the AdventureWorksLTDataMartDatabase that was created above.

6. Change the Name value from Schema1 to AdventureWorksLTDataMartSchema to give the schema a meaningful name.



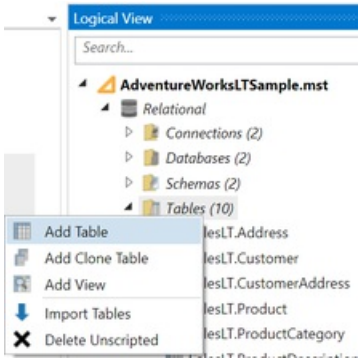
7. Save the project to persist your changes to the project files. See Saving a Project for more information.

Biml supports several types of tables, so when you are creating a new one, your first action will be to choose a type. Here are the types and their uses:

TYPE	USE
Table	Represents a basic relational table
CloneTable	Represents a copy of an existing table, with the ability to add additional columns.
View	Represents a view, and supports additional modeling necessary for defining SQL definition.

Create Table

1. Right click on the Tables node in the logical view, and click Add Table from the resulting menu. A new table will be added with the default name Table_n_, where n equals the count of tables in the model.



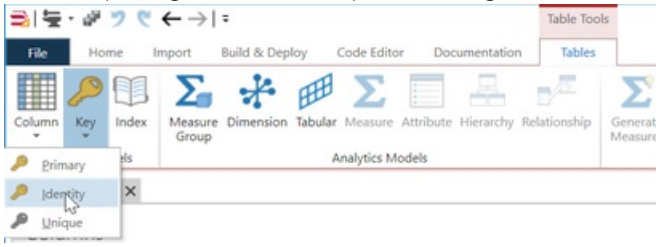
2. Right click on the newly added table node in the logical view, and choose Rename. Enter the new name value (EtlLog in this example) and press Enter to save the change.

3. Select EtlLog in the logical view, and change the Schema drop down in the Properties tool window to AdventureWorksLTDataMartDatabase.AdventureWorksLTDataMartSchema.

   If the Properties tool window is not visible, you can show it by selecting the Home tab and the View split button.

4. Double-click or right-click on EtlLog and choose View Designer to open the designer.

5. Add a column to the table by clicking the **Column** button on the Tables ribbon, alternatively you can also click the **Add Table Column** plus sign. For this example, add a regular Table Column.



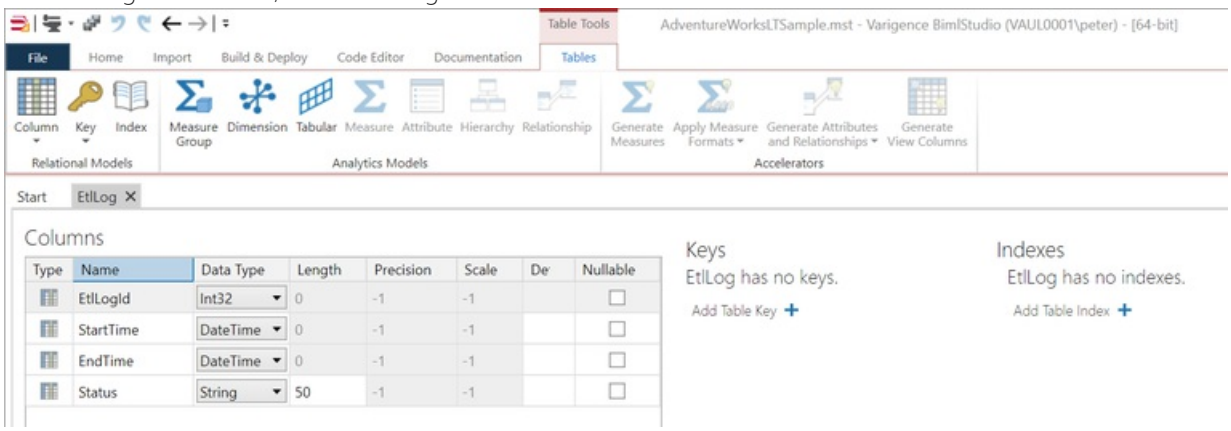
6. Change the column's name from Column1 to EtlLogId by selecting the Column1 value in the grid, and pressing F2 to begin editing. You can also just begin typing in the field.

7. Repeat steps 6 and 7 to add the following columns:

NAME	DATA TYPE	LENGTH
StartTime	DateTime	
EndTime	DateTime	

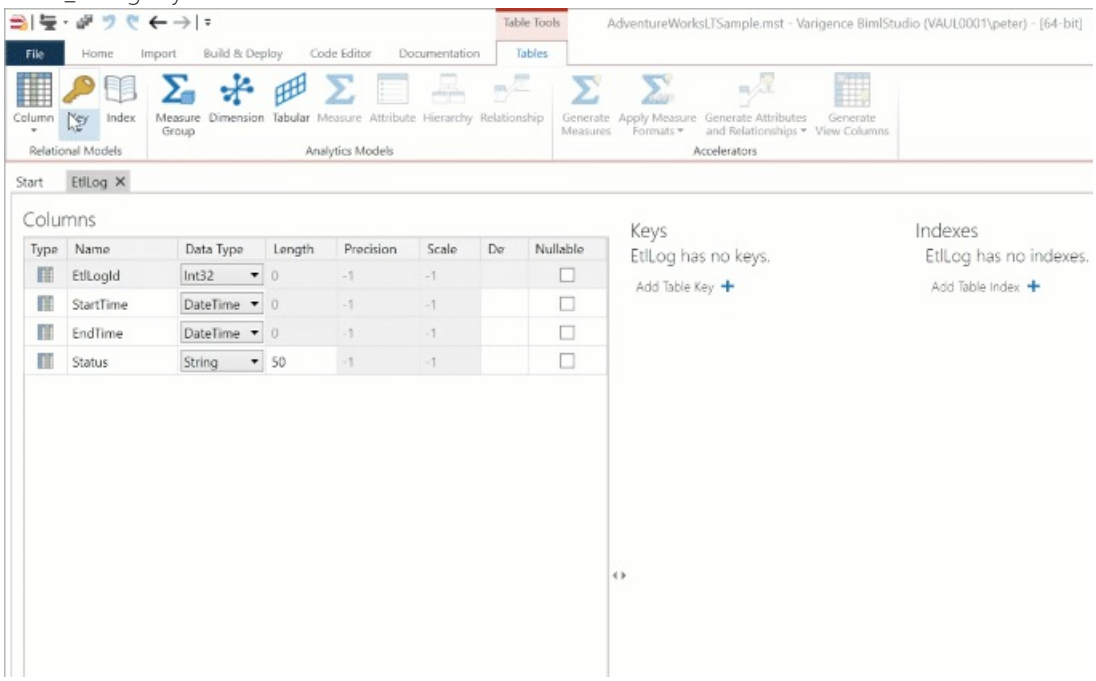
Status	String	50

After adding the columns, the columns grid should look like this:



8. To add an identity key, click the lower portion of the Key split button, and select Identity.

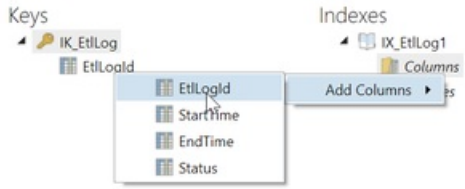
9. Drag and drop the EtlLogId column to the newly created key. Start dragging from the Type column. The cursor will change to black pointer with 4 arrows when your mouse is in the correct location to start a drag and drop operation. Drop the column on the IK_EtlLog key to add the column to it.



Identity Keys are automatically treated as the primary key for a table in BIML. If you need an identity column that is not the primary key, this can presently be done with an alter table script in the CustomExtensions for the table.

10. To add an index, click the Index button in the ribbon. A new index will be added. You may need to scroll the designer window to the right to see the indexes section.

11. Right click on the Columns folder under IX_EtlLog1, and choose Add Columns..EtlLogId.



12. Save the project to persist your changes to the project files.

The table has now been created and is part of the model.

View Biml

If you right click the table and click **View Biml** it should look similar to the following snippet.

```
    <Biml xmlns="http://schemas.varigence.com/biml.xsd">
        <Tables>
            <Table Name="EtlLog" SchemaName="AdventureWorksLTDataMartDatabase.AdventureWorksLTDataMartSchema"
LogicalDisplayFolder="AwDm">
                <Columns>
                    <Column Name="EtlLogId" />
                    <Column Name="StartTime" DataType="DateTime" />
                    <Column Name="EndTime" DataType="DateTime" />
                    <Column Name="Status" DataType="String" Length="50" />
                </Columns>
                <Keys>
                    <Identity Name="IK_EtlLog" Clustered="true">
                        <Columns>
                            <Column ColumnName="EtlLogId" />
                        </Columns>
                    </Identity>
                </Keys>
                <Indexes>
                    <Index Name="IX_EtlLog1">
                        <Columns>
                            <Column ColumnName="EtlLogId" />
                        </Columns>
                    </Index>
                </Indexes>
            </Table>
        </Tables>
    </Biml>
```

Creating a Dimension Table

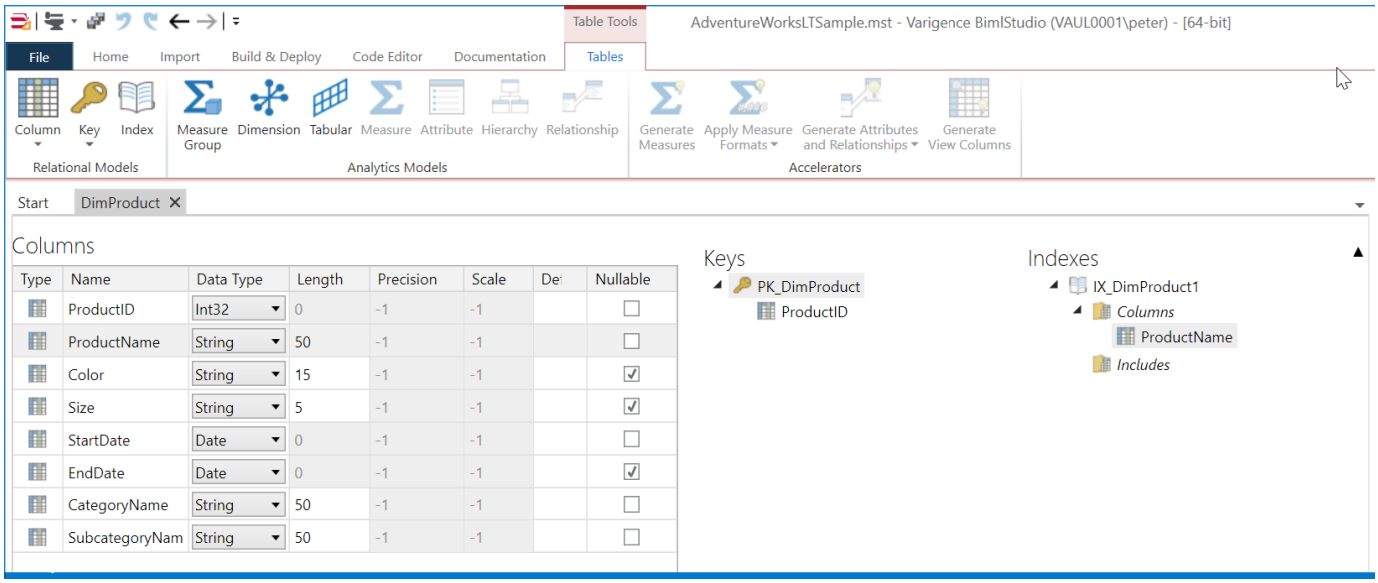
Dimension tables let you model not only the relational parts of a table, but also the analytical aspects. If your dimension is going to be exposed through an OLAP cube, this allows you to model all aspects of the dimension in the same editor. The relational features are modeled the same way for all table types, while the analytical features vary depending on the purpose of the table. For dimensions, the dimension editor lets you model the properties that control how a Microsoft Analysis Services dimension will be emitted.

To create a dimension, follow these steps:

1. Follow the directions from the Creating a Basic Table topic to define the relational portion of the dimension, using this information:
   Table Name: DimProduct **Schema:** AdventureWorksLTDataMartSchema **Keys:** IK_DimProduct (using column ProductID)
   Indexes: IX_DimProduct_ProductName (using column ProductName) **Columns:**

NAME	DATA TYPE	LENGTH	NULLABLE
ProductID	Int32	No	
ProductName	String	50	No
Color	String	15	Yes
Size	String	5	Yes
StartDate	Date	No	
EndDate	Date	Yes	
CategoryName	String	50	No
SubcategoryName	String	50	No

   The table editor should look like this:



2. To add the dimension attributes click **Dimension** on the **Table Tools** ribbon.

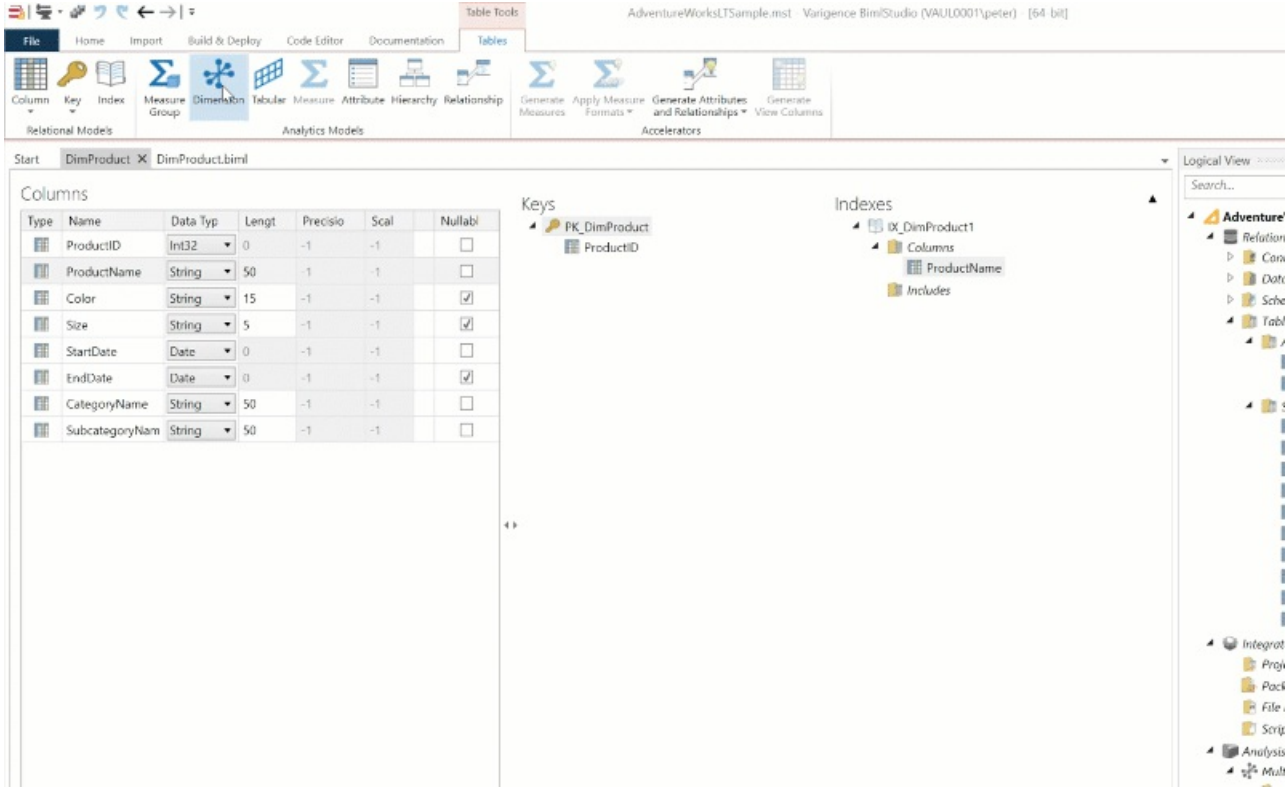
3. The Attributes area lets you define the attributes that will appear in the OLAP dimension. You can populate this by dragging and

dropping columns from the columns grid into the Attributes area. Drag the ProductID column to the Attributes area. Notice that when you select an attribute, you can view and edit all its properties in the Property Grid.

4. Right click on the Product attribute to set the attribute usage to Key.

5. Rename the attribute to Product Name using the context menu or by pressing F2 with the attribute selected.

6. Drag and drop the ProductName column to the Name Column folder item under the Product Name attribute.



7. You can create additional attributes by using an Accelerator. On the Table Tools ribbon, Tables tab, click the Generate Attributes and Relationships split button. Choosing the Generate Attributes and Relationships menu item will automatically create attributes for appropriate columns from the table, and relationships for those attributes.

8. The attributes and relationships will be populated

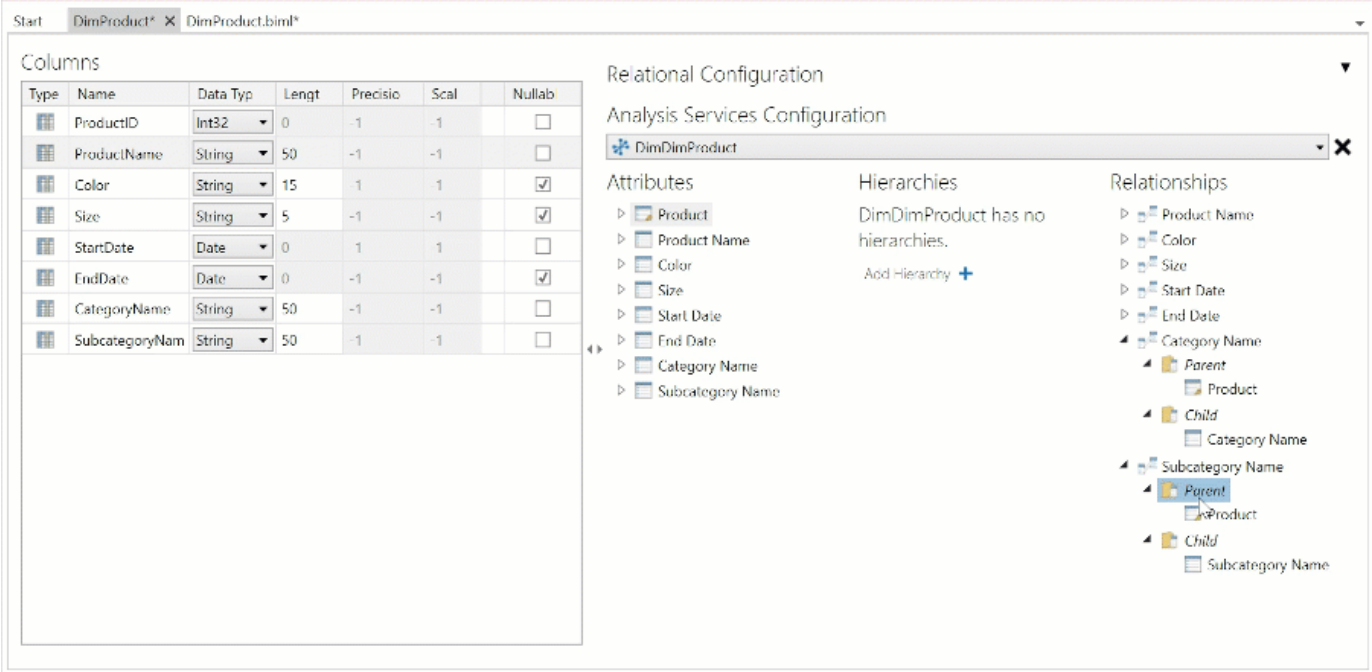
   Note that the accelerator populates a default set of relationships, ensuring that every attribute is related to the key.

   It's always a good idea to review the relationships to ensure they match your data. Attribute relationships are important to getting the best performance from Analysis Services.

9. To adjust the relationships to reflect the data, locate the Category Name relationship in the Relationships area. Right click on the Parent folder and select Set Parent Attribute. Select the Subcategory Name attribute.

   This creates a relationship chain that includes Product Name -> Subcategory Name -> Category Name.

10. Next, you will add a hierarchy. Right click in the empty Hierarchies area, and select Add Hierarchy.



11. Rename the hierarchy to Categories. You can add levels to the hierarchy by right-clicking it and choosing Add Attributes from the context menu, or by dragging and dropping the attributes from the Attributes tree view. Add the following attributes (in order): Category Name, Subcategory Name, and Product Name. The Hierarchies area should look like this when you are finished: Drag and drop the attribute on the Hierarchy node (Categories) to add the attribute as a new level at the end. Dropping an attribute on an existing level will replace the level's associated attribute. You can also drag and drop levels within a hierarchy.

12. Save the project to commit your changes.

Once a dimension is created in the model, it will emit as a relational table. If it is referenced from a cube, it will also emit as an Analysis Services dimension.

View Biml

If you right click the table and click **View Biml** it should look similar to the following snippet.

```
    <Biml xmlns="http://schemas.varigence.com/biml.xsd">
        <Tables>
            <Table Name="DimProduct"
SchemaName="AdventureWorksLTDataMartDatabase.AdventureWorksLTDataMartSchema" LogicalDisplayFolder="AwDm">
                <Columns>
                    <Column Name="ProductID" DataType="Int32" IsNullable="false" />
                    <Column Name="ProductName" DataType="String" Length="50" IsNullable="false" />
                    <Column Name="Color" DataType="String" Length="15" IsNullable="true" />
                    <Column Name="Size" DataType="String" Length="5" IsNullable="true" />
                    <Column Name="StartDate" DataType="Date" IsNullable="false" />
                    <Column Name="EndDate" DataType="Date" IsNullable="true" />
                    <Column Name="CategoryName" DataType="String" Length="50" IsNullable="false" />
                    <Column Name="SubcategoryName" DataType="String" Length="50" IsNullable="false" />
                </Columns>
                <Keys>
                    <PrimaryKey Name="PK_DimProduct" Clustered="true">
                        <Columns>
                            <Column ColumnName="ProductID" />
                        </Columns>
                    </PrimaryKey>
                </Keys>
                <Indexes>
                    <Index Name="IX_DimProduct1">
                        <Columns>
                            <Column ColumnName="ProductName" />
                        </Columns>
```

```
                    </Index>
                </Indexes>
                <AnalysisMetadata>
                    <Dimension Name="DimDimProduct">
                        <Attributes>
                            <Attribute Name="Product" Usage="Key">
                                <KeyColumns>
                                    <SnowflakeKeyColumn ColumnName="ProductID" />
                                </KeyColumns>
                                <NameColumn ColumnName="ProductName" />
                            </Attribute>
                            <Attribute Name="Product Name">
                                <KeyColumns>
                                    <KeyColumn ColumnName="ProductName" />
                                </KeyColumns>
                            </Attribute>
                            <Attribute Name="Color">
                                <KeyColumns>
                                    <KeyColumn ColumnName="Color" />
                                </KeyColumns>
                            </Attribute>
                            <Attribute Name="Size">
                                <KeyColumns>
                                    <KeyColumn ColumnName="Size" />
                                </KeyColumns>
                            </Attribute>
                            <Attribute Name="Start Date">
                                <KeyColumns>
                                    <KeyColumn ColumnName="StartDate" />
                                </KeyColumns>
                            </Attribute>
                            <Attribute Name="End Date">
                                <KeyColumns>
                                    <KeyColumn ColumnName="EndDate" />
                                </KeyColumns>
                            </Attribute>
                            <Attribute Name="Category Name">
                                <KeyColumns>
                                    <KeyColumn ColumnName="CategoryName" />
                                </KeyColumns>
                            </Attribute>
                            <Attribute Name="Subcategory Name">
                                <KeyColumns>
                                    <KeyColumn ColumnName="SubcategoryName" />
                                </KeyColumns>
                            </Attribute>
                        </Attributes>
                        <Relationships>
                            <Relationship Name="Product Name" ParentAttributeName="Product"
ChildAttributeName="Product Name" />
                            <Relationship Name="Color" ParentAttributeName="Product"
ChildAttributeName="Color" />
                            <Relationship Name="Size" ParentAttributeName="Product" ChildAttributeName="Size"
/>
                            <Relationship Name="Start Date" ParentAttributeName="Product"
ChildAttributeName="Start Date" />
                            <Relationship Name="End Date" ParentAttributeName="Product"
ChildAttributeName="End Date" />
                            <Relationship Name="Category Name" ParentAttributeName="Product"
ChildAttributeName="Category Name" />
                            <Relationship Name="Subcategory Name" ParentAttributeName="Category Name"
ChildAttributeName="Subcategory Name" />
                        </Relationships>
                        <AttributeHierarchies>
                            <Hierarchy Name="Categories">
                                <Levels>
```

```
                            <Levels>
                                <Level Name="Category Name" AttributeName="Category Name" />
                                <Level Name="Subcategory Name" AttributeName="Subcategory Name" />
                                <Level Name="Product Name" AttributeName="Product Name" />
                            </Levels>
                        </Hierarchy>
                    </AttributeHierarchies>
                </Dimension>
            </AnalysisMetadata>
        </Table>
    </Tables>
</Biml>
```

Adding existing files to a project

It's possible to add existing Biml files to a project using the Project View. This is useful if you want to reuse Biml from a previous project, or you are sharing a common Biml file between projects.

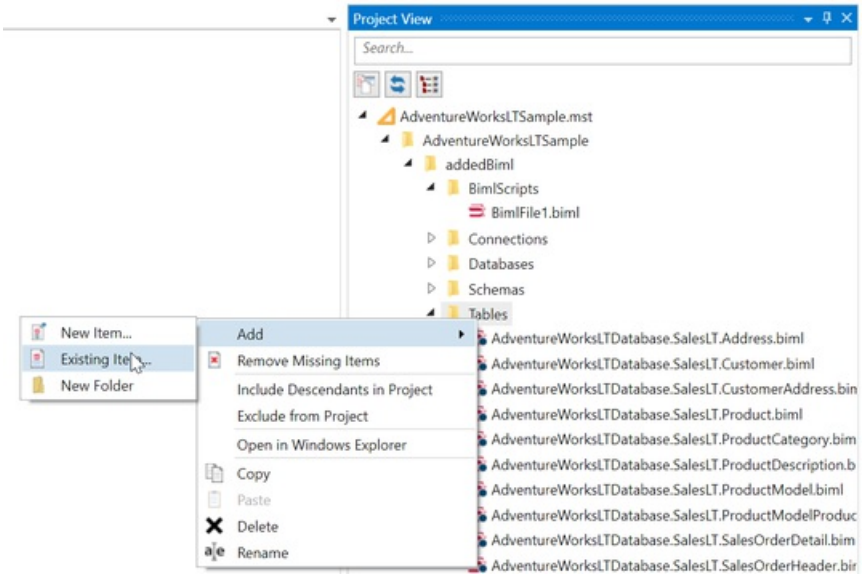
To add existing Biml files to a project, follow these steps:

1. (Optional but recommended) Using Windows Explorer, copy the Biml files to the appropriate location in the project structure. For dimensions, this would be [project folder]\addedBiml\Tables

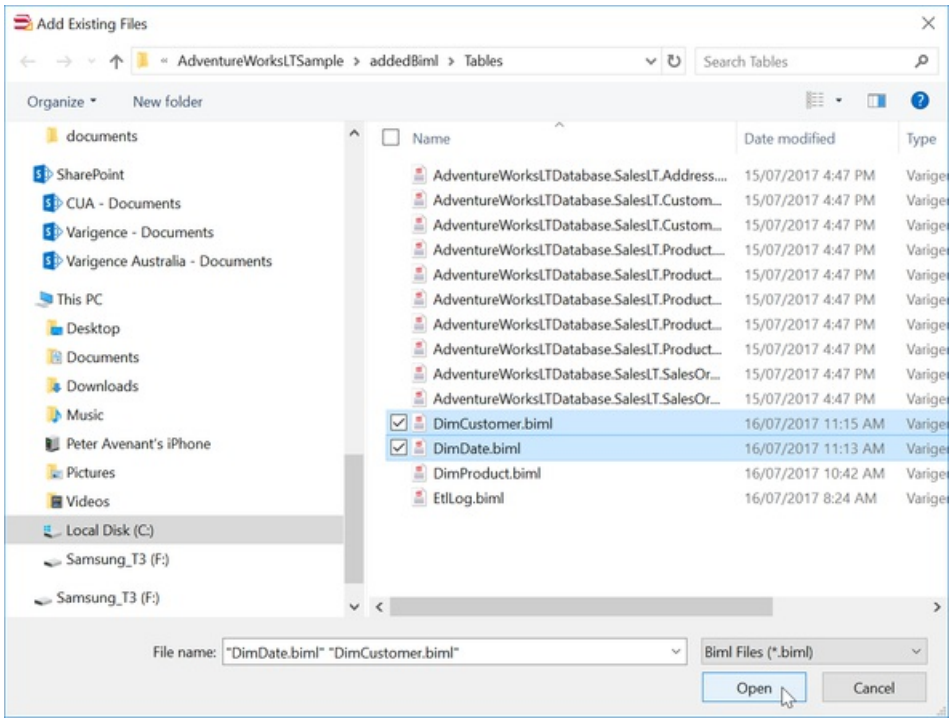
   Files do not have to be located under the project folder to be added, but it does make management of Biml files easier.

2. Select the Project View tool window. It normally appears as a tab beside the Logical View tool window at the right side of the BimlStudio application window. If it is not visible, use the View button on the Home tab of the Ribbon to display it.

3. Expand the tree and Right-click the Tables item in the tree view, and choose Add > Existing Item...

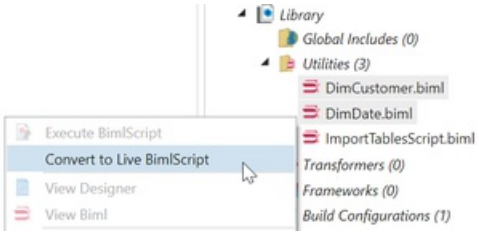


1. In the Add Existing Files dialog, locate the folder that contains the Biml files. Select the files that you want to add. Multiple files can be selected from this dialog.



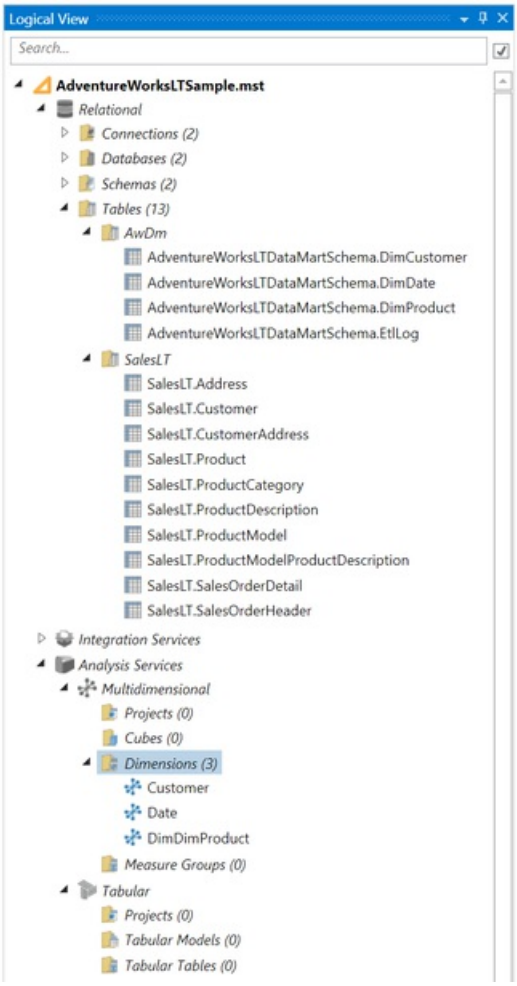
2. Click **Open** to add the files to the project.

3. Using the **Logcal View** you might need to **Convert to Live BimlScript** the files.



4. Save to project to commit your changes.

For the AdventureWorks LT sample, you will need to download a zip file containing two Biml files: DimDate and DimCustomer for Here. After unzipping the file add them to your project. After adding the files, the logical view should include the DimDate and DimCustomer dimensions.


Logical View
Search...
AdventureWorksLTSample.mst
Relational
Connections (2)
Databases (2)
Schemas (2)
Tables (13)
AwDm
AdventureWorksLTDataMartSchema.DimCustomer
AdventureWorksLTDataMartSchema.DimDate
AdventureWorksLTDataMartSchema.DimProduct
AdventureWorksLTDataMartSchema.EtlLog
SalesLT
SalesLT.Address
SalesLT.Customer
SalesLT.CustomerAddress
SalesLT.Product
SalesLT.ProductCategory
SalesLT.ProductDescription
SalesLT.ProductModel
SalesLT.ProductModelProductDescription
SalesLT.SalesOrderDetail
SalesLT.SalesOrderHeader
Integration Services
Analysis Services
Multidimensional
Projects (0)
Cubes (0)
Dimensions (3)
Customer
Date
DimDimProduct
Measure Groups (0)
Tabular
Projects (0)
Tabular Models (0)
Tabular Tables (0)

Creating a Fact Table

Fact tables let you model the relational parts of a table, and the analytical aspects. If your fact is going to be exposed through an OLAP cube, this allows you to model all aspects of the measure group in the same editor. The relational features are modeled the same way for all table types, while the analytical features vary depending on the purpose of the table. For facts, you can model the properties that control how a Microsoft Analysis Services measure group will be emitted through the Dimension editor.

To create a fact, follow these steps:

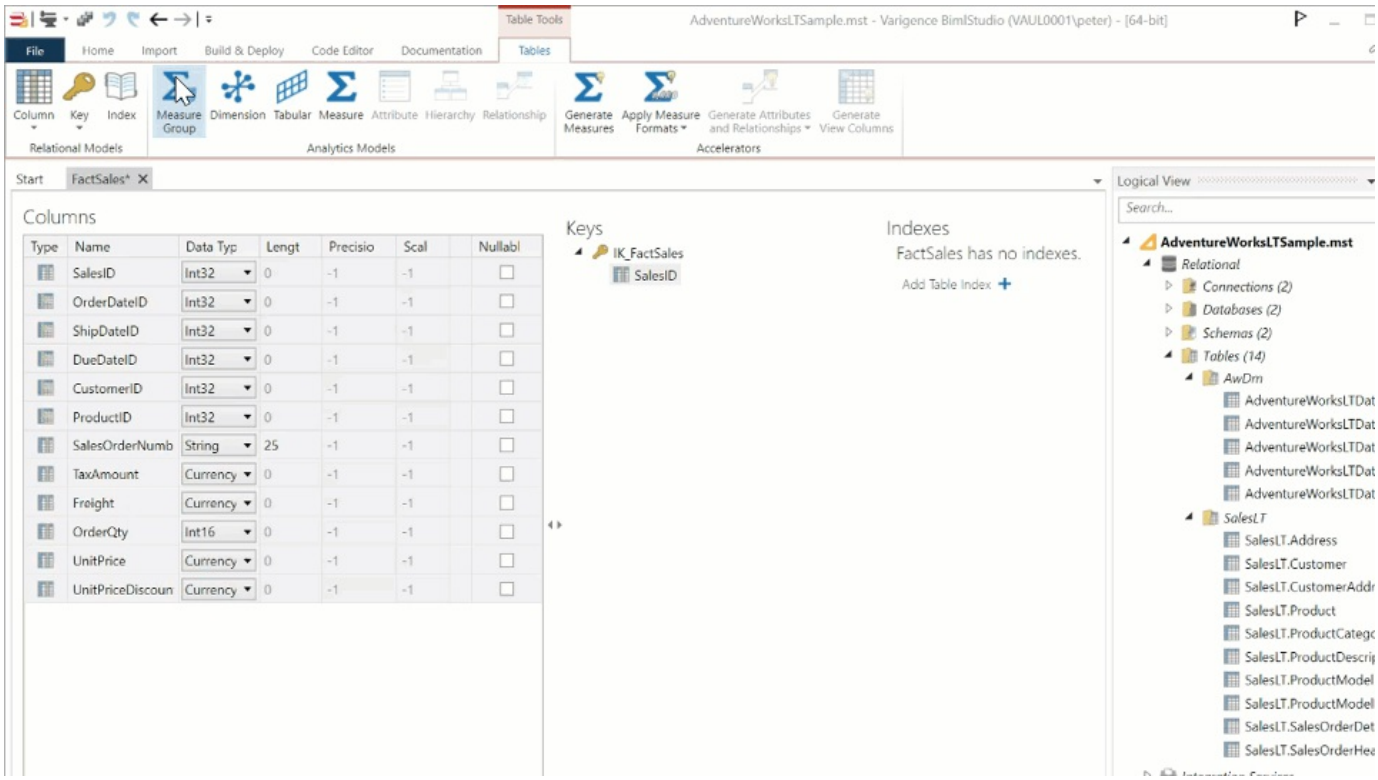
1. Add a new table and rename it to FactSales. For more information about creating the relational portions of fact table using the designer, see Creating a Basic Table.

2. To create the relational portion of the table, you will edit the Biml directly. Right click the in the logical view and choose View Biml.

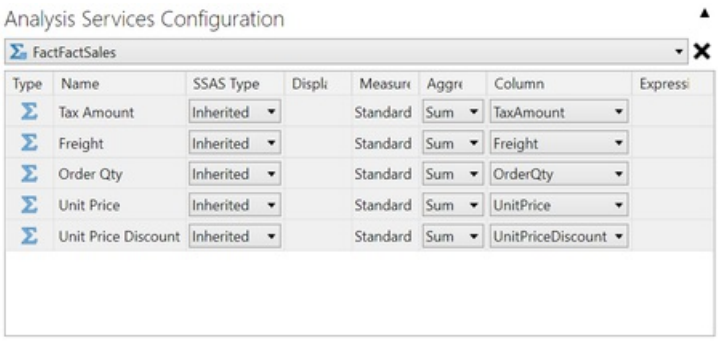
3. Select all the contents of the Biml editor. You can do this by right-clicking in the Biml editor and clicking the Select All menu item. Then delete the contents by pressing the Delete key. Finally, paste in the following Biml:

```
        <Biml xmlns="http://schemas.varigence.com/biml.xsd">
            <Tables>
                <Table Name="FactSales"
SchemaName="AdventureWorksLTDataMartDatabase.AdventureWorksLTDataMartSchema" LogicalDisplayFolder="AwDm">
                    <Columns>
                        <Column Name="SalesID" />
                        <TableReference TableName="DimDate" Name="OrderDateID"
ForeignKeyNameOverride="FK_FactSales_DimDate_OrderDateID" ForeignKeyConstraintMode="DoNotCreate" />
                        <TableReference TableName="DimDate" Name="ShipDateID"
ForeignKeyNameOverride="FK_FactSales_DimDate_ShipDateID" ForeignKeyConstraintMode="DoNotCreate" />
                        <TableReference TableName="DimDate" Name="DueDateID"
ForeignKeyNameOverride="FK_FactSales_DimDate_DueDateID" ForeignKeyConstraintMode="DoNotCreate" />
                        <TableReference TableName="DimCustomer" Name="CustomerID"
ForeignKeyNameOverride="FK_FactSales_DimCustomer_CustomerID" ForeignKeyConstraintMode="DoNotCreate" />
                        <TableReference TableName="DimProduct" Name="ProductID"
ForeignKeyNameOverride="FK_FactSales_DimProduct_ProductID" ForeignKeyConstraintMode="DoNotCreate" />
                        <Column Name="SalesOrderNumber" DataType="String" Length="25" />
                        <Column Name="TaxAmount" DataType="Currency" />
                        <Column Name="Freight" DataType="Currency" />
                        <Column Name="OrderQty" DataType="Int16" />
                        <Column Name="UnitPrice" DataType="Currency" />
                        <Column Name="UnitPriceDiscount" DataType="Currency" />
                    </Columns>
                    <Keys>
                        <Identity Name="IK_FactSales">
                            <Columns>
                                <Column ColumnName="SalesID" />
                            </Columns>
                        </Identity>
                    </Keys>
                </Table>
            </Tables>
        </Biml>
```

1. To create the measures for this fact table, click Measure Group from the Table Tools ribbon, then click Generate Measures. This will populate the measures grid with default measures based on the columns defined in the table.



2. The following measures should be generated:



3. Save the project to persist your changes.

Once a fact is created in the model, it will emit as a relational table. If it is referenced from a cube, it will also emit as an Analysis Services measure group.

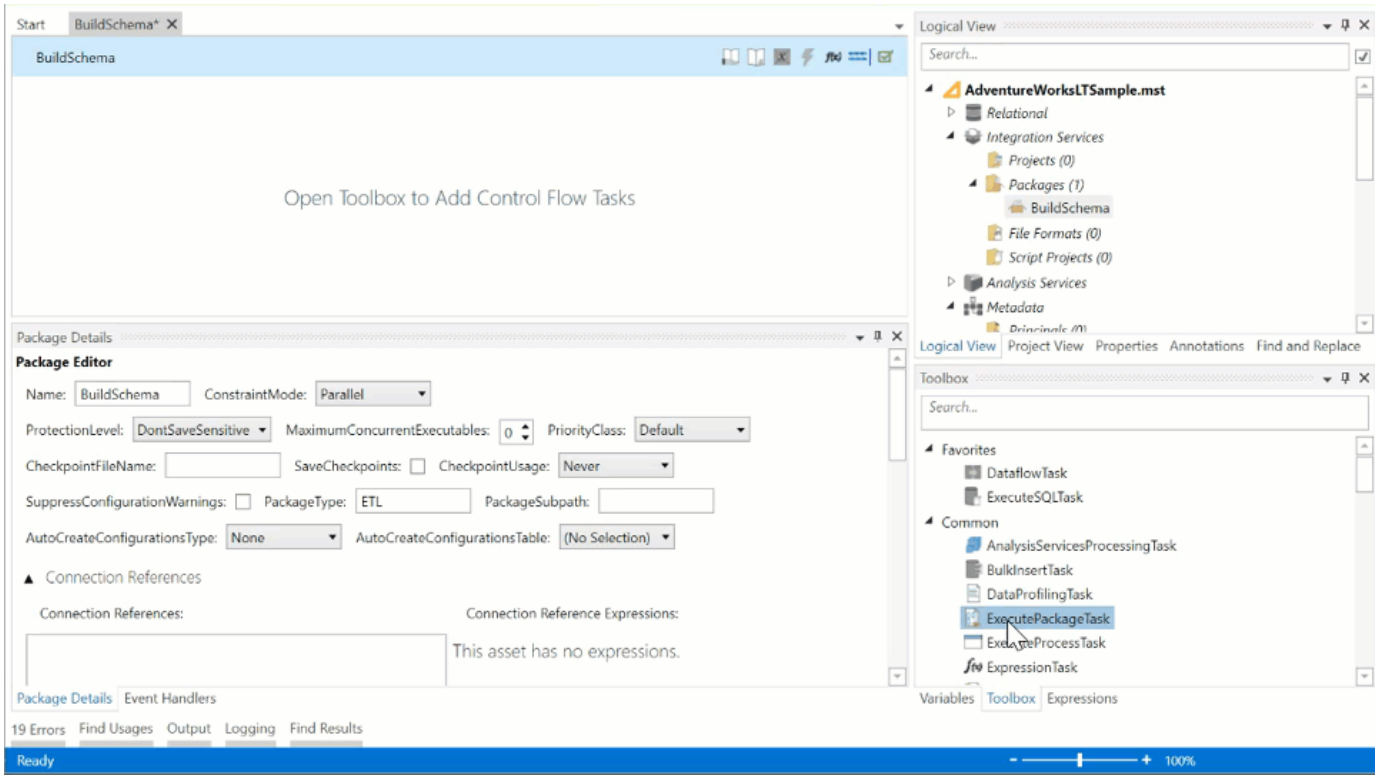
Deploying Tables to SQL

When building and deploying a cube, newly created relational tables need to be deployed to a SQL database. A package, using SSIS ExecutePackage tasks, can accomplish this.

Begin by adding a package to the project. Refer to the first 4 steps in the Creating a Package topic to create a package and rename it to *BuildSchema*.

To deploy the tables to SQL:

1. Find the Toolbox tool window. Drag and drop an ExecutePackage task onto the designer surface.
2. In the Package Details tool window, choose the DimCustomer tab in the Table dropdown. Additionally, for clarity, rename this task to be DimCustomer.



3. Repeat steps 1 and 2 for the following tables:

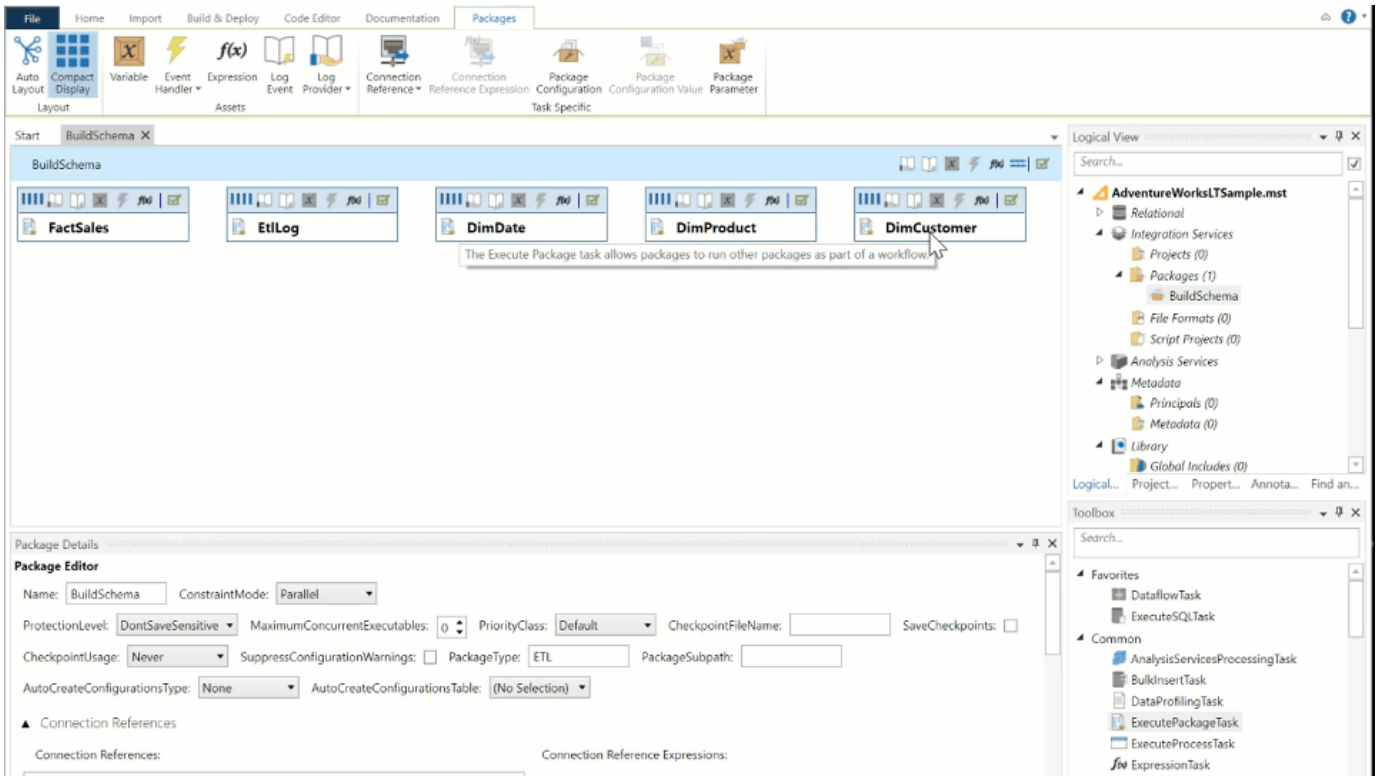
   - DimDate
   - DimProduct
   - EtlLog
   - FactSales

   When finished, the package editor should look as follows:

4. To create a precedence constraint, drag a line from the bottom anchor of the DimCustomer ExecutePackage task to the top anchor of the DimDate ExecutePackage task.

   Start dragging from the bottom anchor of the DimCustomer task. While dragging, a blue line will follow the mouse cursor. Move the mouse cursor over the the DimCustomer task's top anchor and release.

5. Create edges so that the tasks are linked together as follows:



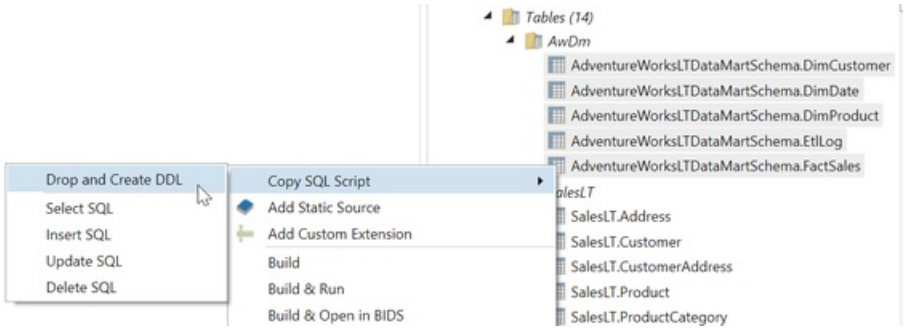
6. Ensure that your project settings have **Use Project Deployment** un-selected. For more information on Use Project Deployment, see Configuring Project Settings.

7. Right click on the package and select Build & Run. This executes the tasks in the package, deploying the tables to the AdventureWorksLTDataMart database.

Alternative Deployment

You can also deploy the tables manually by selecting all the tables.

1. Right click then **Copy SQL Script**, then **Drop and Create DDL**.



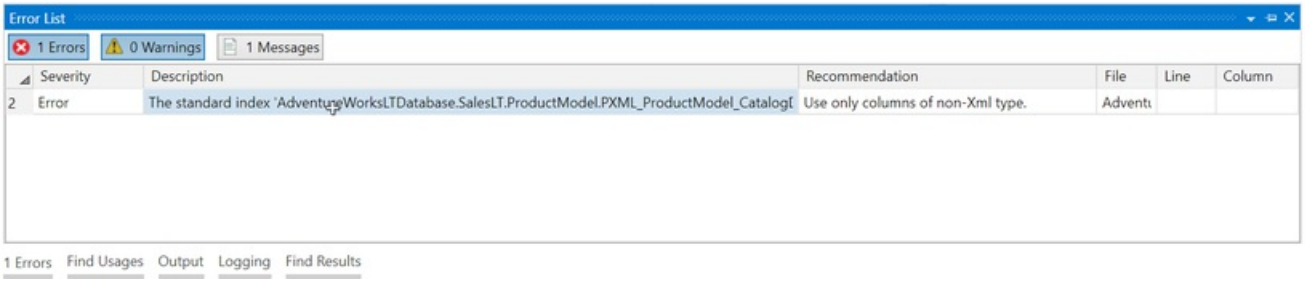
2. The SQL will be in your clipboard and can be pasted into Microsoft SQL Server Management Studio and executed.

Preparing to Build the Example Project

At this point, all the tables necessary for the example project have been modeled. This is a good point to build the project, so that you can verify what's been produced so far. There are several steps involved in building the project: verifying the physical setup of the project, checking for design time errors, and verifying project settings.

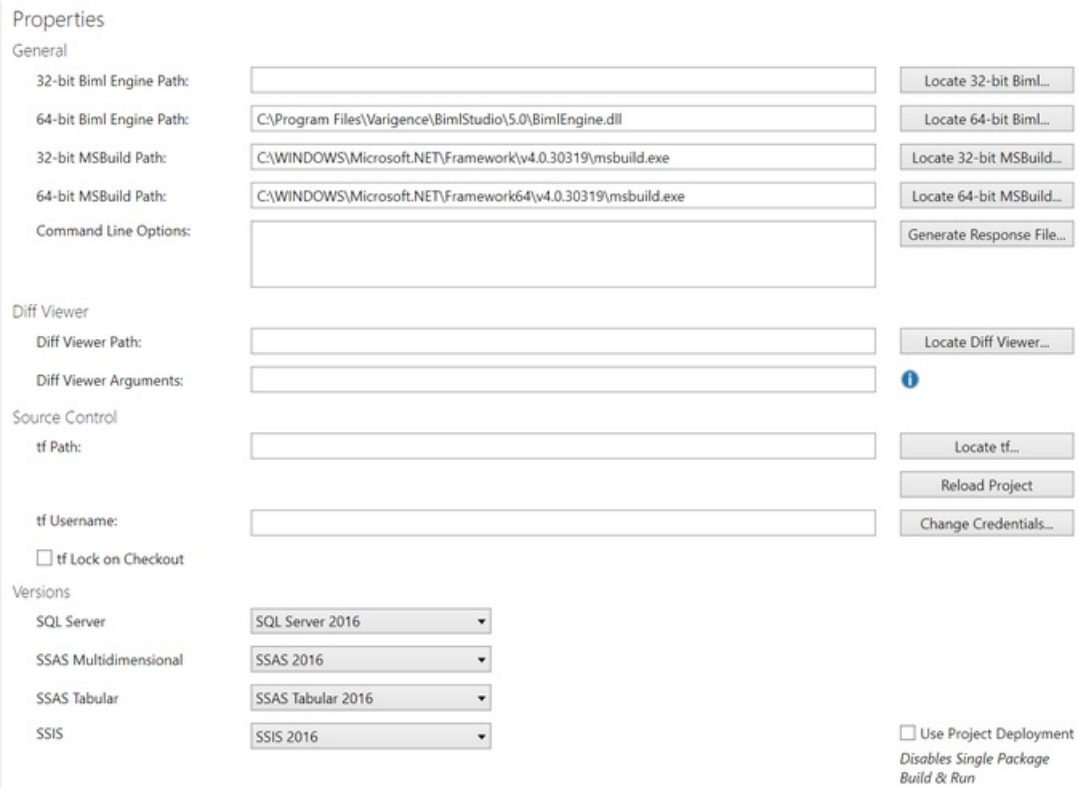
Checking for Errors

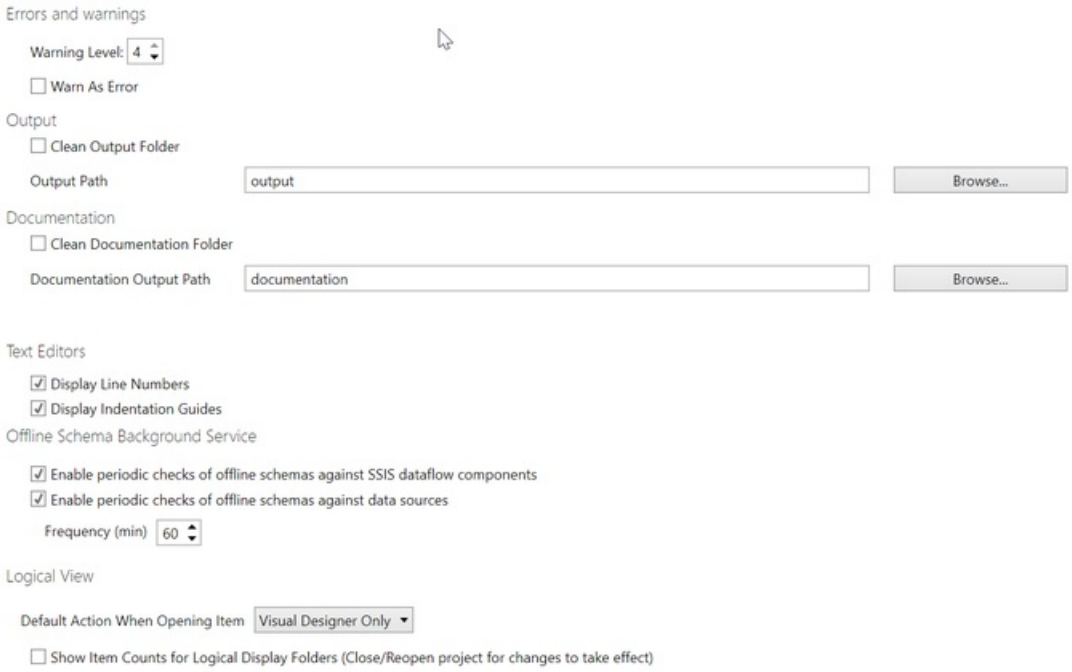
The Error List tool window displays errors. By default, it is displayed in a tab at the bottom of the BimlStudio window. If it is not visible, you can show it by selecting the Error List from the View button on the Home tab of the ribbon. The Error List shows 3 categories of messages. Errors will prevent a build from succeeding. Warnings are issues that should be addressed, but will not prevent a successful build. Messages provide suggestions for improvements, and will not prevent a successful build. You can filter each category of message by clicking the buttons at the top of the Error List. The example project should only have one item in the error list, a warning. This warning comes from the source tables that you imported in an earlier step. You can safely ignore it, since the source tables are only included for metadata. If there are any errors or other warnings, review the preceding steps to see if there are any differences.



Verify Project Settings

The project settings allow you to control various aspects of the project and build process. This includes the version of the SQL Server technologies that you are targeting, the location of the Hadron compiler, and the output folder for the compiled BI assets. To review the project settings, see Configuring Project Settings. For the example project, make sure that all versions are targeting your current version of SQL Server. Also confirm that the Warn As Error check box is unchecked. Your project settings should look similar to this, but with the appropriate version of the SQL Server technologies selected:





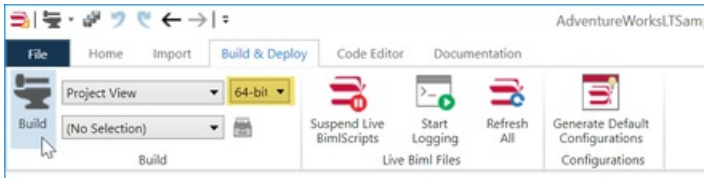
Once you have verified the project settings and that there are no errors, you are ready to build the project.

Building the Example Project

Building a project produces compiled output for each item included in an emittable file.

To build a project:

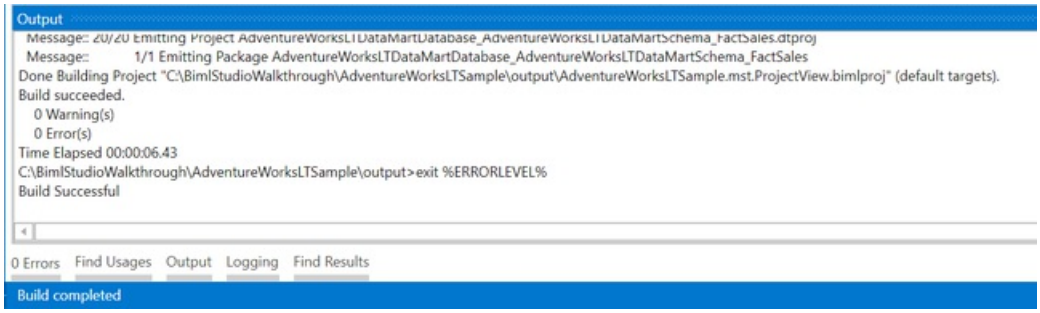
1. On the **Build & Deploy** tab, click the Build button.



2. Using the 32-bit or 64-bit version of BimlStudio. For the example, choose the version that matches your BimlStudio installation. You can determine whether BimlStudio is running in 32-bit or 64-bit by looking in the title bar of theBimlStudioMist window. If you see [64-bit], then BimlStudio is running as a 64-bit process. Otherwise, it's running in 32-bit.

3. The build will start. You can see its progress in the Output tool window, normally located in the lower portion of the BimlStudio window.

4. Scroll to the bottom of the Output tool window to see the results of the build. You should see values similar to this:



If the build completed with no errors, you can check the project's output folder to see what was produced.

From this point, you can create a cube within BimlStudio and build out its actions, KPIs, etc... When finished, rebuild the BimlStudio project to emit a dwproj that can be opened in Visual Studio for cube deployment.

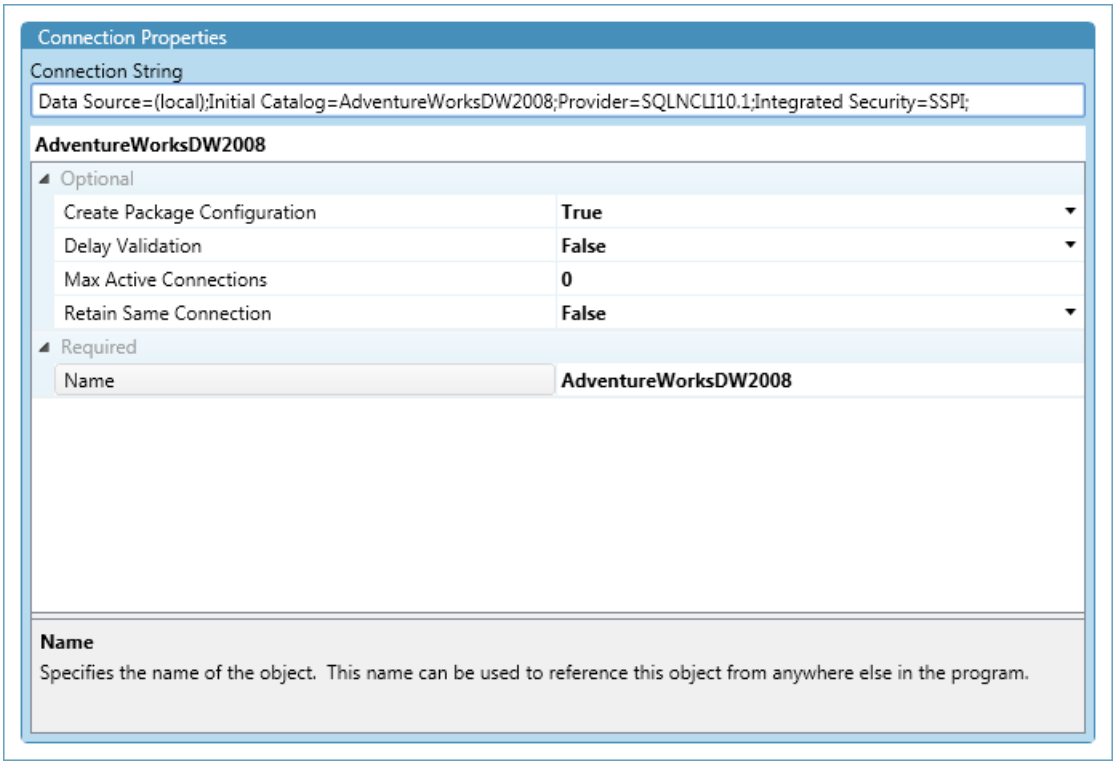
Download Solution

You can download the full solution from here.

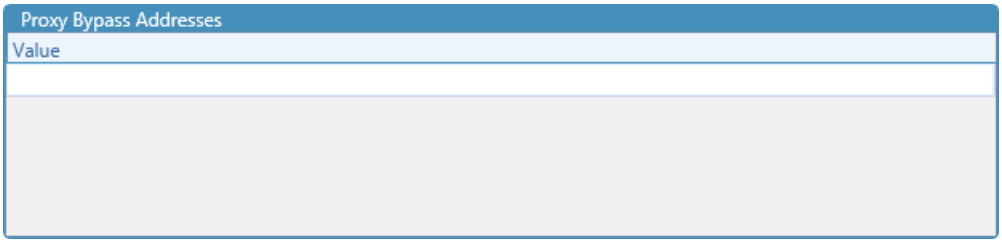
Connection Editor

Use the Connections editor to edit properties and connection strings.

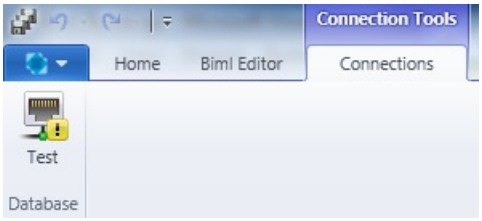
For database connections, enter a connection string and provide authentication information.



For HTTP connections, proxy bypass addresses can also be entered.



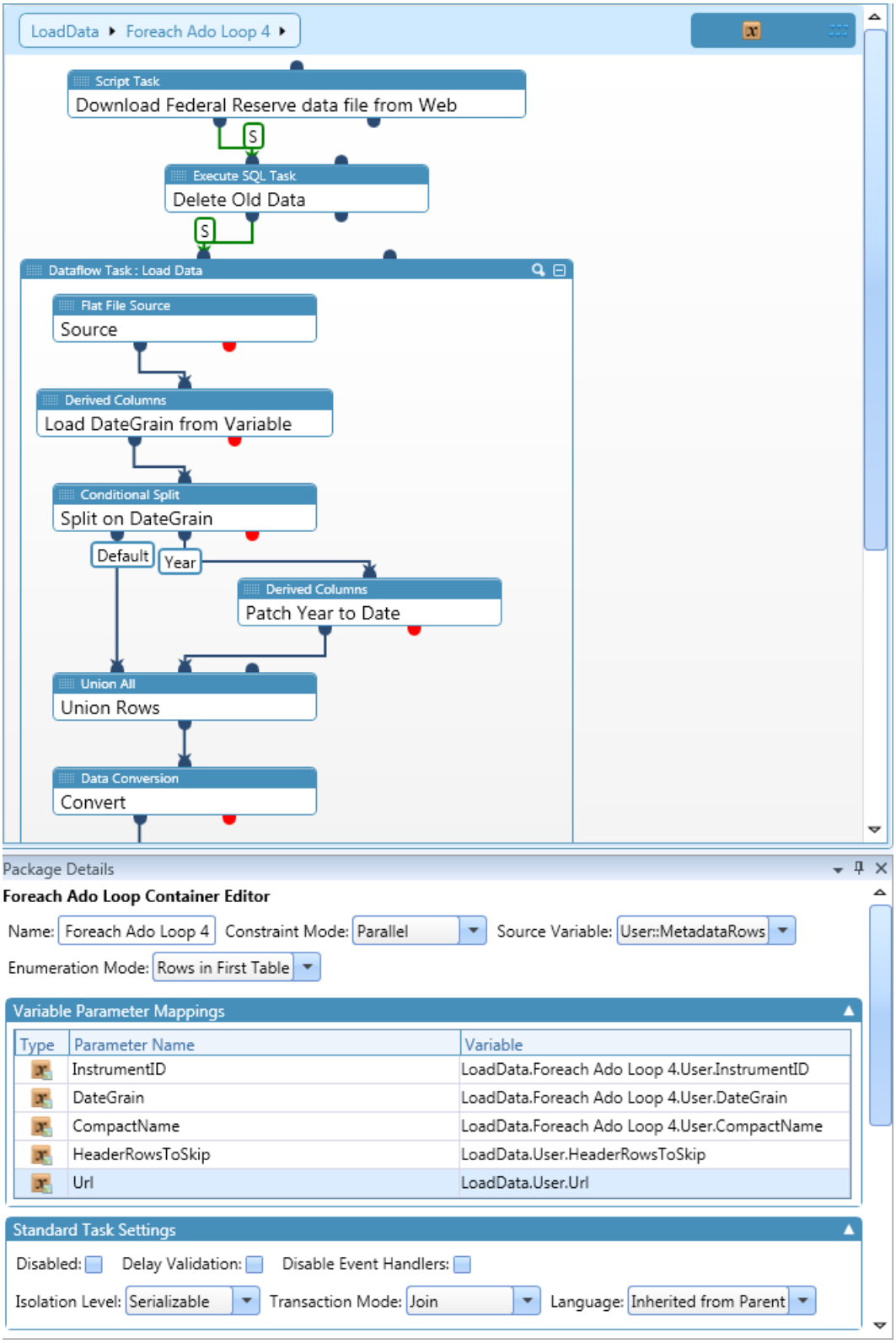
The connection editor's ribbon provides the following functions:



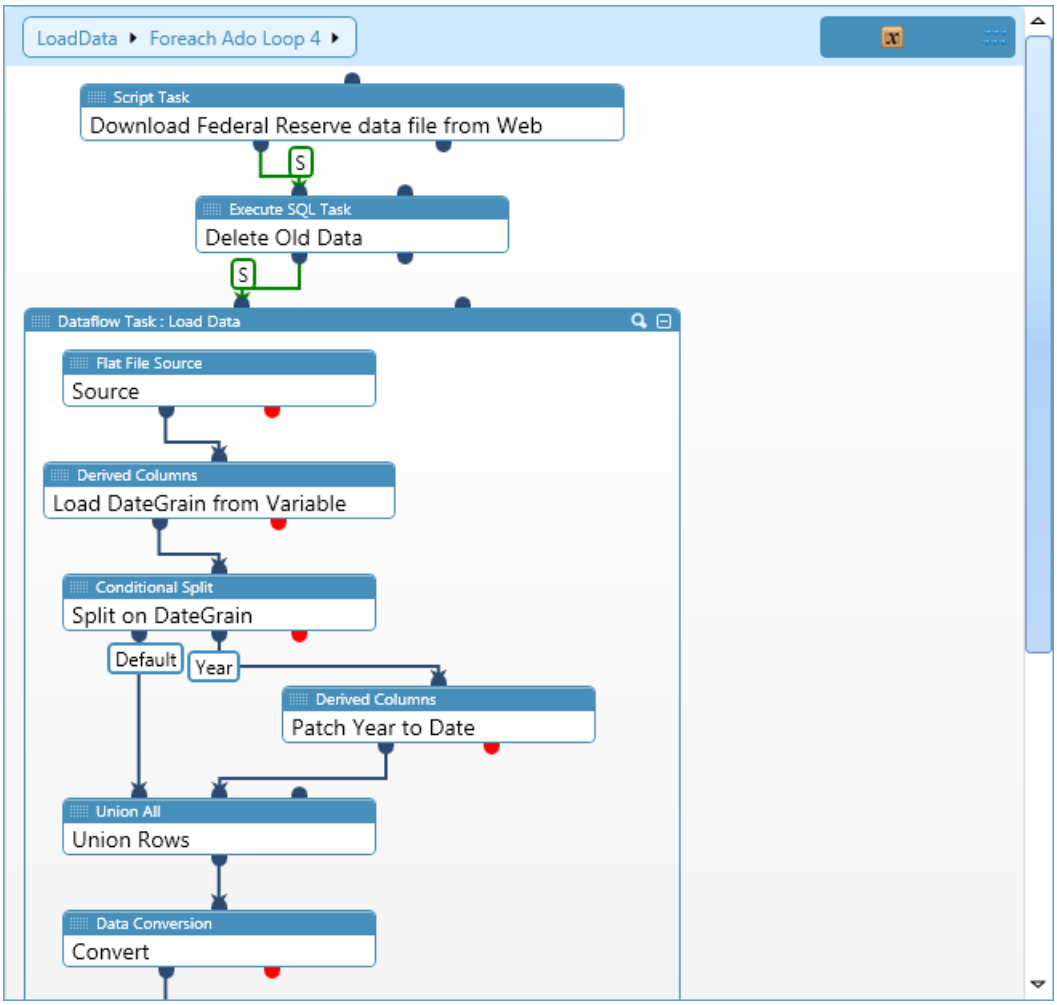
BUTTON	DESCRIPTION
Test	Test OLE DB based connections to confirm they work.

Package Editor

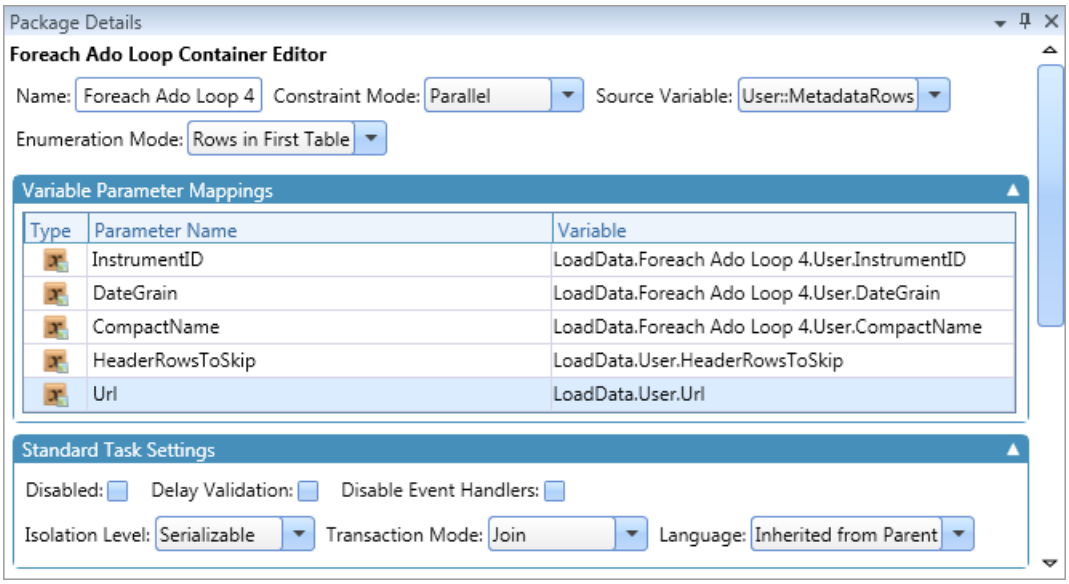
Use the package editor to construct and edit a control flow in a package.



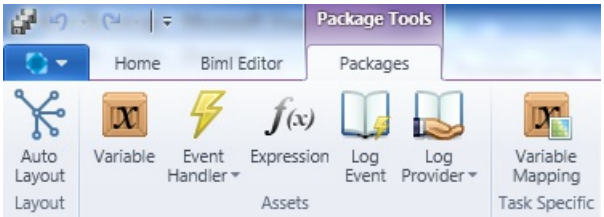
Add tasks that support data flow, data preparation, perform BI functions, and execute scripts. Add containers that can repeat workflows or divide a control flow into subsets. Connect containers and tasks into an ordered control flow using precedence constraints.



Edit a task or transform in the Package Details tool window.



The package editor's ribbon provides the following common functions:

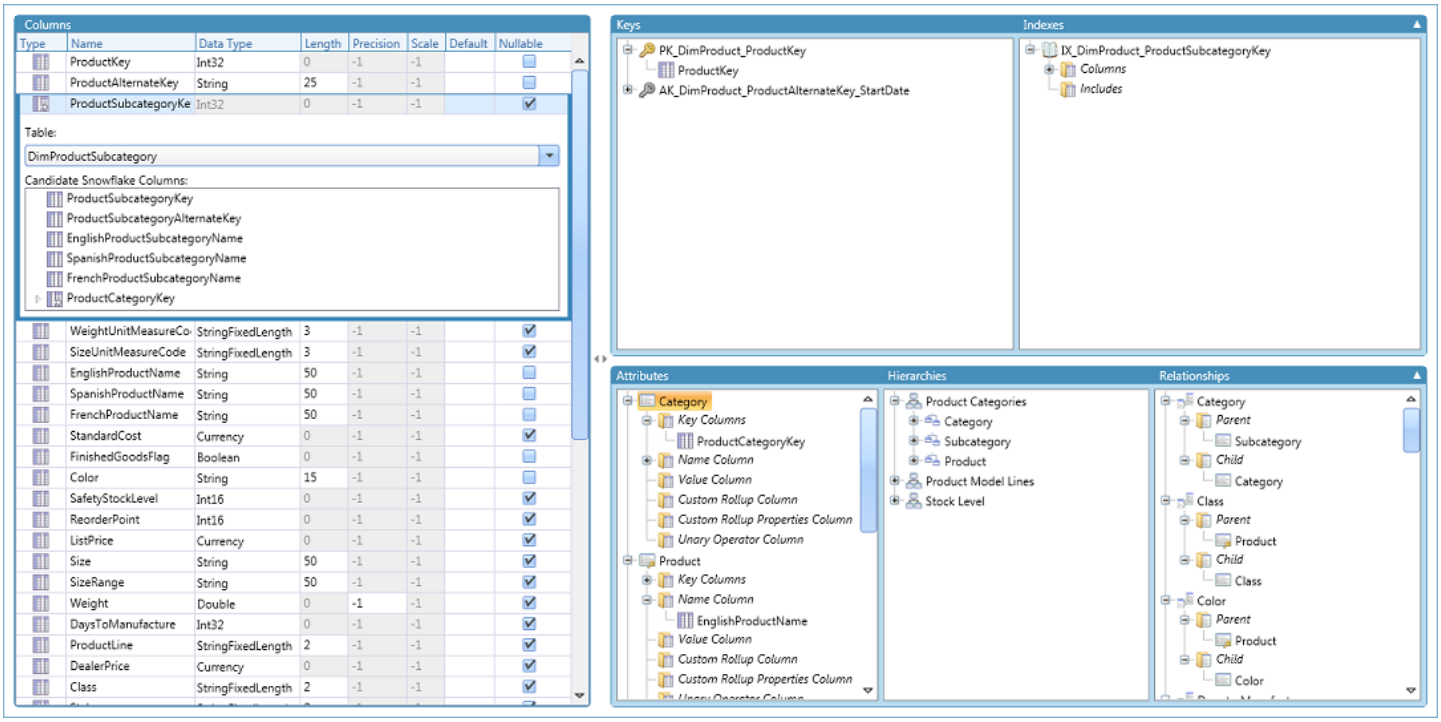


BUTTON	DESCRIPTION
Auto Layout	Rearrange nodes and edges in the designer for better spacing and organization.
Variable	Add a variable to a selected task or transformation.
Event Handler	Choose an event handler to add to a selected task or transformation.
Expression	Add an expression to a selected task or transformation.
Log Event	Add a log event to a selected task or transformation.
Log Provider	Add a log provider to a selected task or transformation.

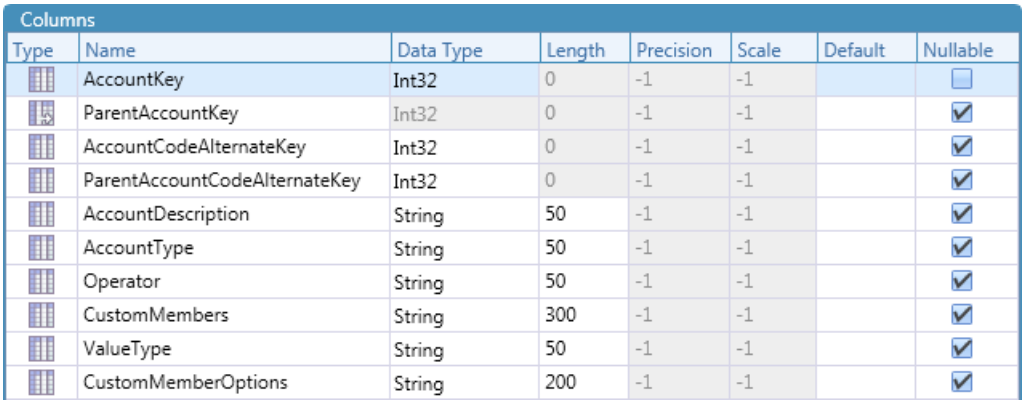
Additionally, buttons are dynamically added for features that are task or transform specific.

Table Editor

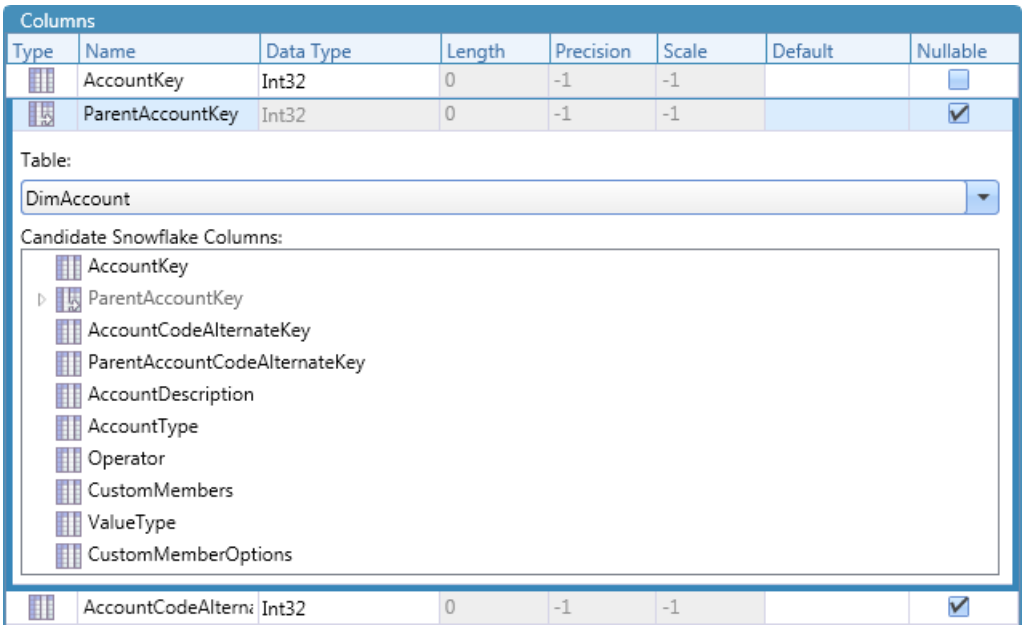
Use the Table editor to edit tables, by adding, changing, or removing columns, keys, indexes, and fact or dimension specific objects.



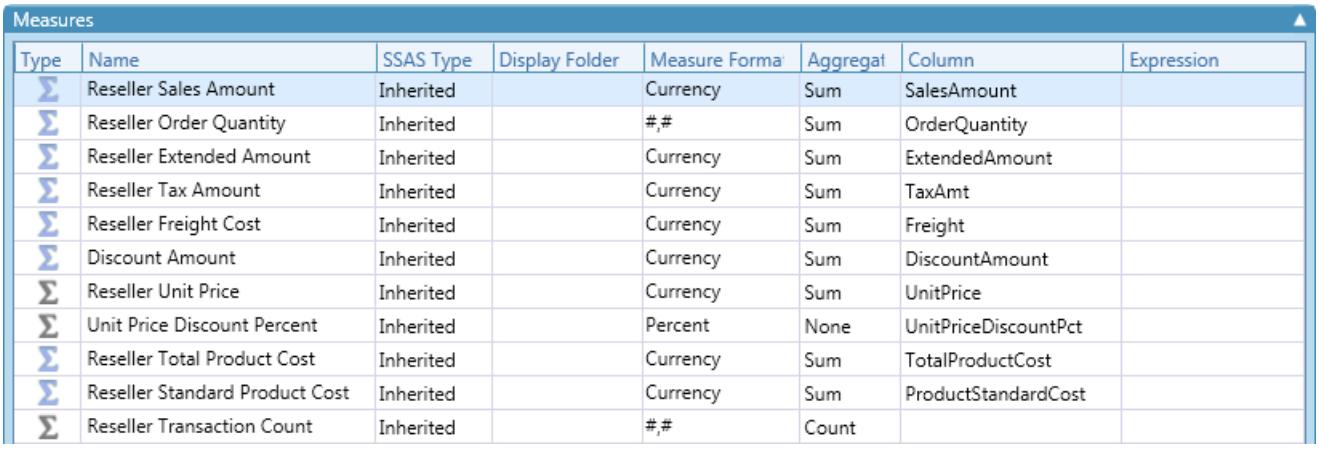
Enter a column's name, data type, length, precision, and other properties in the column's data grid row. Drag and drop a column to create related keys, indexes, or attributes.



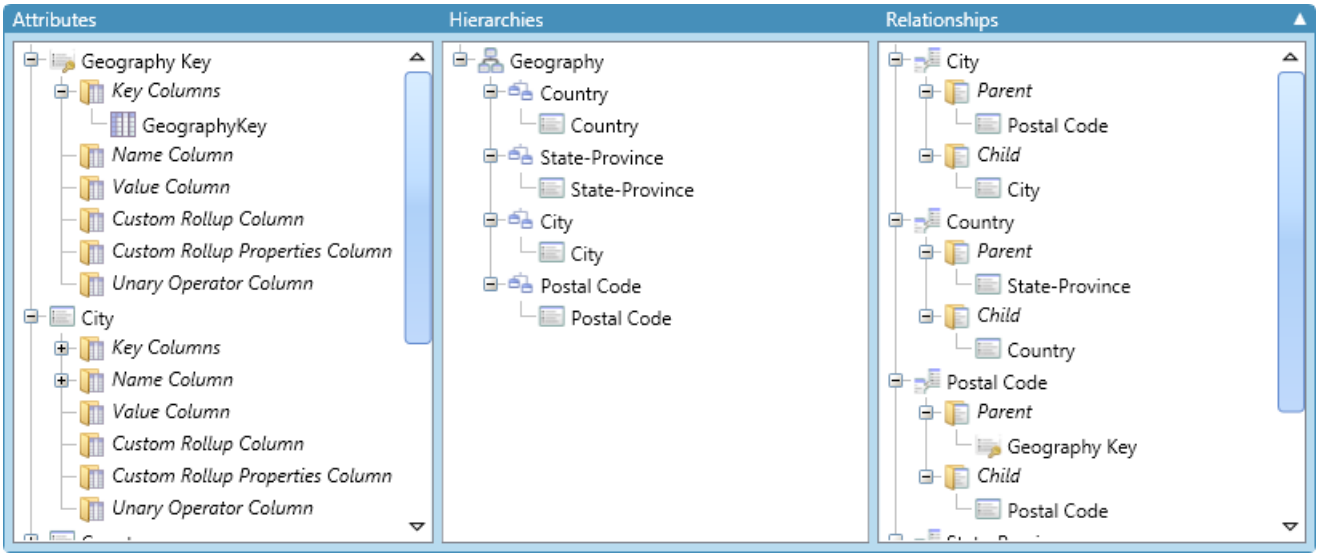
Edit a table reference column's table and snowflake columns in its data grid row's details area. Drag and drop a snowflake column to create related keys, indexes, or attributes.



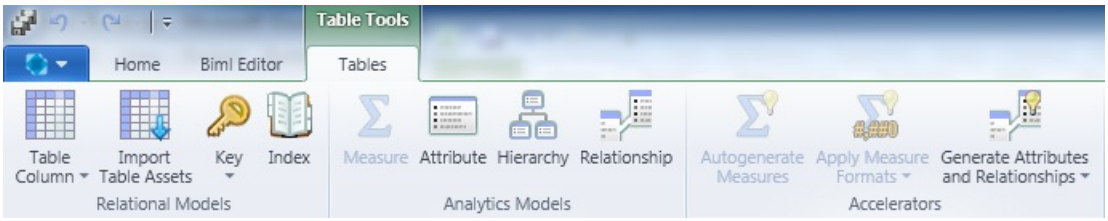
Edit a measure's name, SSAS type, measure format, aggregation type, and other properties in its data grid row.



Edit a dimension's attributes, hierarchies, and relationships directly in their tree views.



The table editor's ribbon provides the following functions:



BUTTON	DESCRIPTION
Table Column	Choose and add table columns to a table. The default is table column.
Import Table Assets	Display the Import Tables dialog, for importing schemas and tables from a database.
Key	Choose a key to add to a table. The default is primary key; if a primary key is present, then the default is unique key.
Index	Add an index to a table.
Measure	Add a measure to a fact table.
Attribute	Add an attribute to a dimension.
Hierarchy	Add a hierarchy to a dimension.
Relationship	Add a relationship to a dimension.
Autogenerate Mesures	Add a measure for each table column not already associated with a measure.
Apply Measure Formats	Choose to preserve or overwrite existing measure formats.
Generate Attribute and Relationships	Choose to generate attributes, relationships, or both for a dimension.

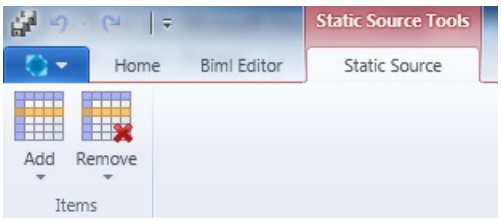
Static Source Editor

Use the Static Source editor to add and edit static source rows.

	HeightID	MetricHeightName	ImperialHeightName
1	-1	N'Unknown'	N'Unknown'
2	0	N'Unknown'	N'Unknown'
3	80	N'80 cm'	N'2ft 11in'
4	90	N'90 cm'	N'2ft 0in'
5	91	N'91 cm'	N'2ft 0in'
6	92	N'92 cm'	N'3ft 1in'
7	93	N'93 cm'	N'3ft 1in'
8	94	N'94 cm'	N'3ft 1in'
9	95	N'95 cm'	N'3ft 2in'
10	96	N'96 cm'	N'3ft 2in'
11	97	N'97 cm'	N'3ft 2in'
12	98	N'98 cm'	N'3ft 3in'
13	99	N'99 cm'	N'3ft 3in'
14	100	N'100 cm'	N'3ft 4in'
15	101	N'101 cm'	N'3ft 4in'
16	102	N'102 cm'	N'3ft 4in'
17	103	N'103 cm'	N'3ft 5in'
18	104	N'104 cm'	N'3ft 5in'
19	105	N'105 cm'	N'3ft 6in'
20	106	N'106 cm'	N'3ft 6in'

If added columns are needed, add table columns and table reference columns within the editor.

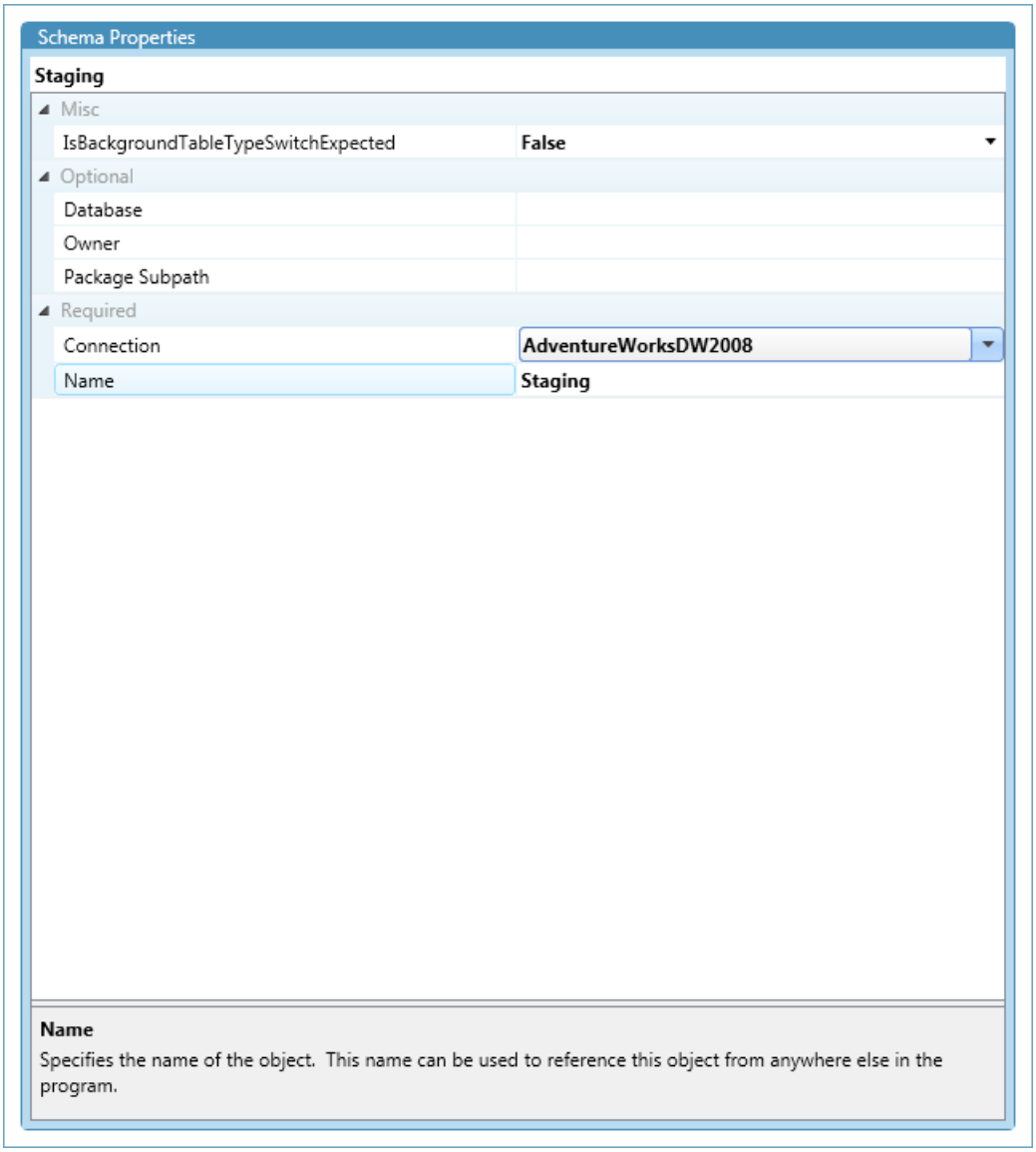
The static source editor's ribbon provides the following functions:



BUTTON	DESCRIPTION
Add	Add a row to the static source, or a table or table reference column to the table. The default is adding a row.
Remove	Remove a selected row or column from the static source.

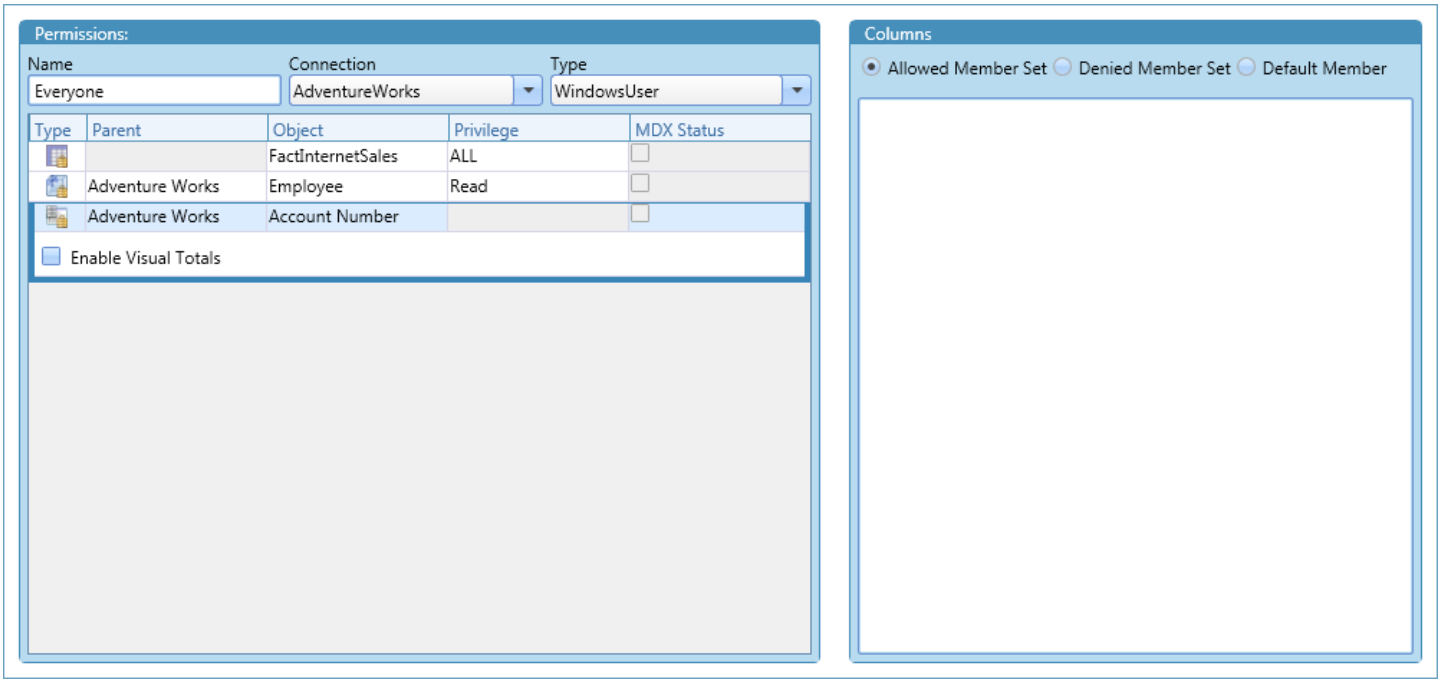
Schema Editor

Use the Schema editor to edit schema properties.

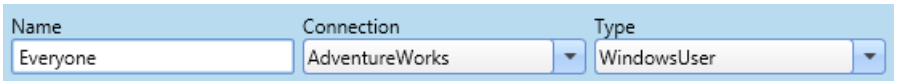


Principal Editor

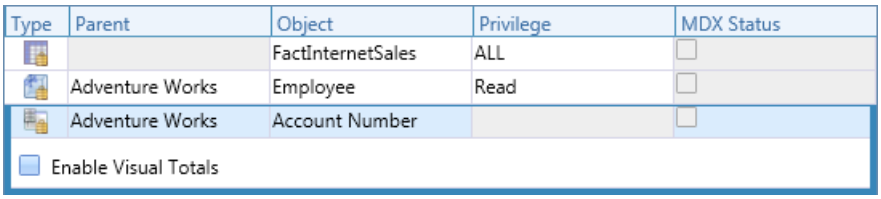
Use the Principal editor to edit a principal, and add and edit permissions for a principal.



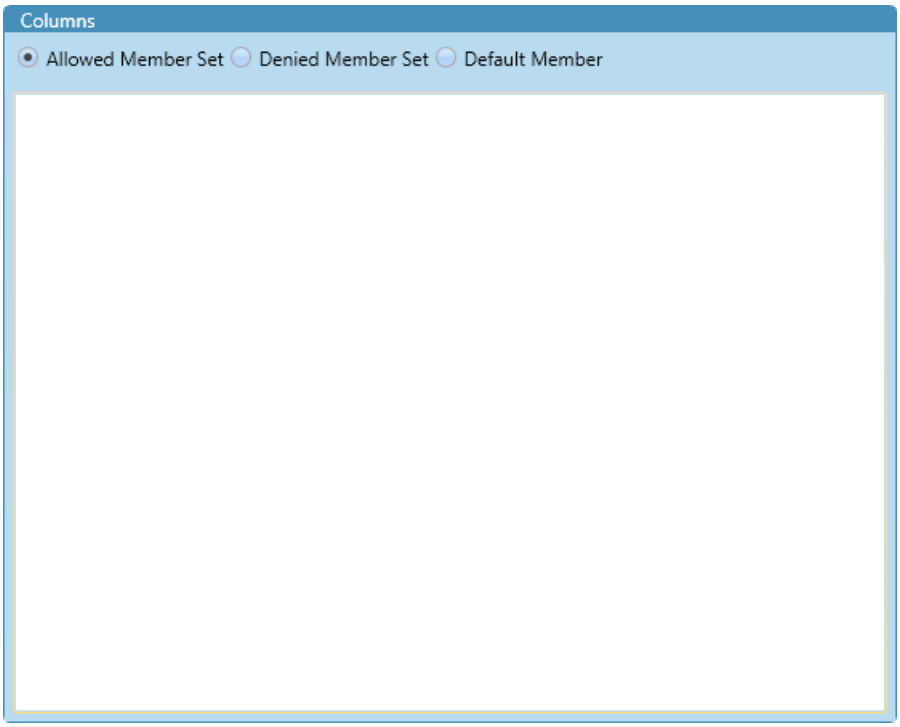
Enter a principal's name, connection, and type above its data grid.



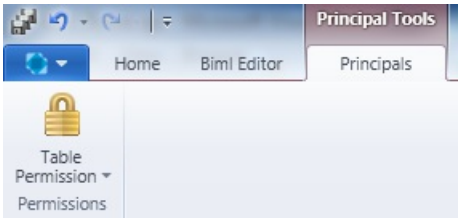
Edit a permission's applicable parent, object, privilege, and MDX status in its data grid row. Permission specific properties are controlled in the row's details area.



For Cube Dimension Attribute, Database Dimension Attribute, and Cube Cells permissions, select an expression type and enter an expression in a pop-out text editor.



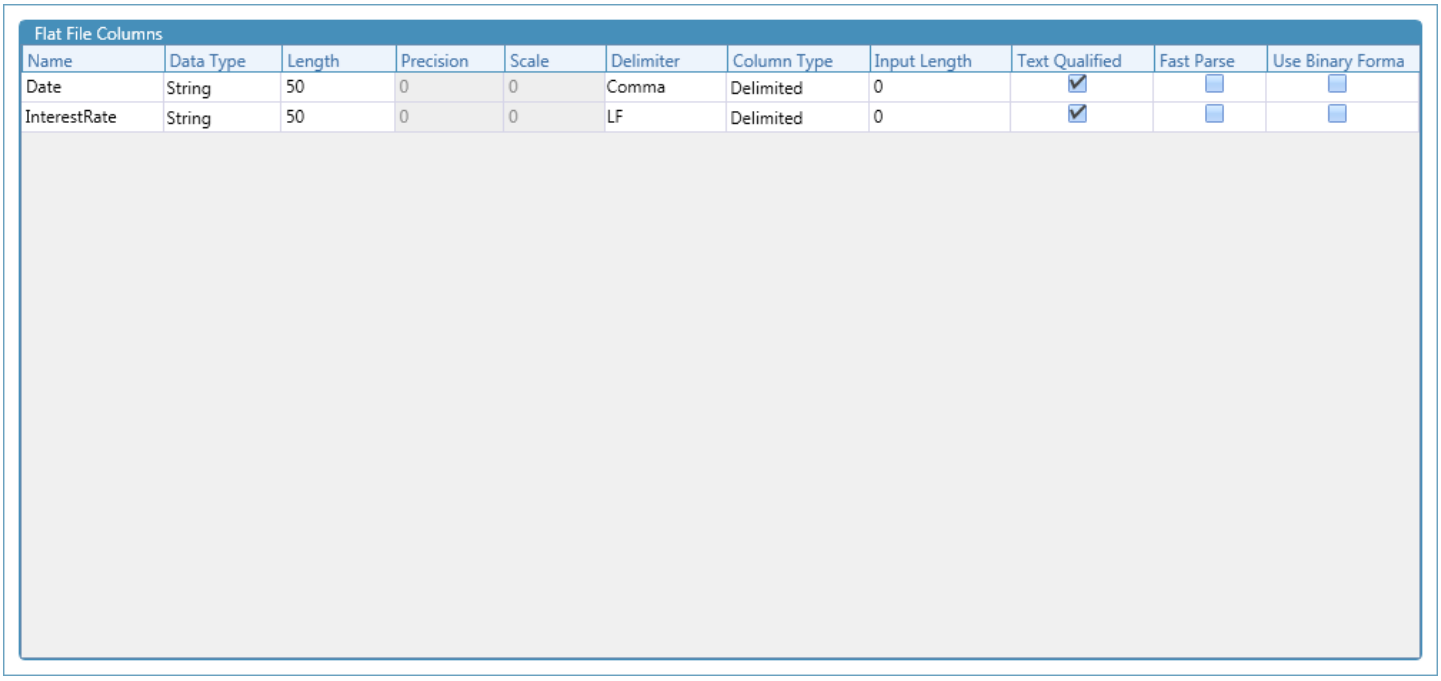
The principal editor's ribbon provides the following functions:



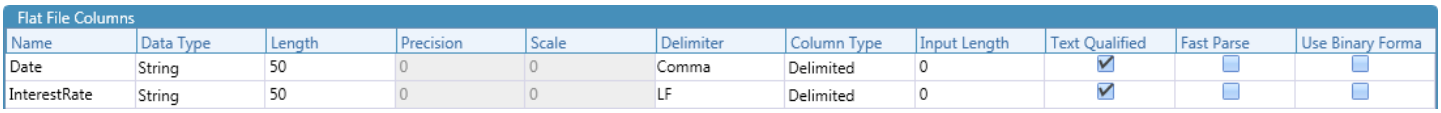
BUTTON	DESCRIPTION
Table Permission	Choose and add a permission to the principal. The default is table permission.

File Format Editor

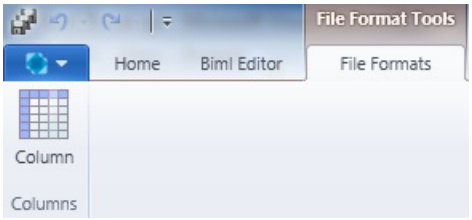
Use the File Format editor to define the columns in a file format.



Enter a column's name, data type, delimiter, column type, and other properties in its data grid row.



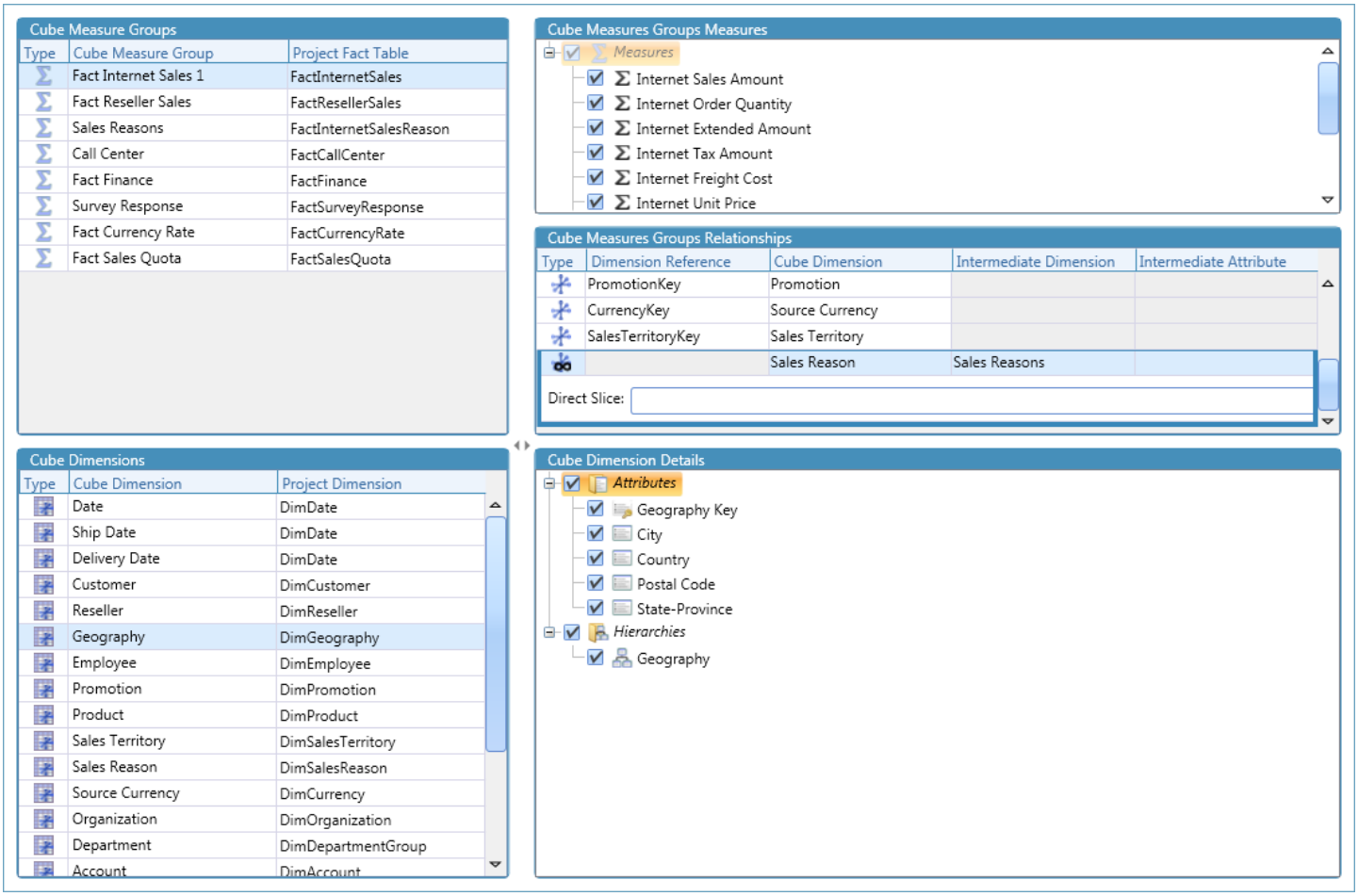
The file format editor's ribbon provides the following functions:



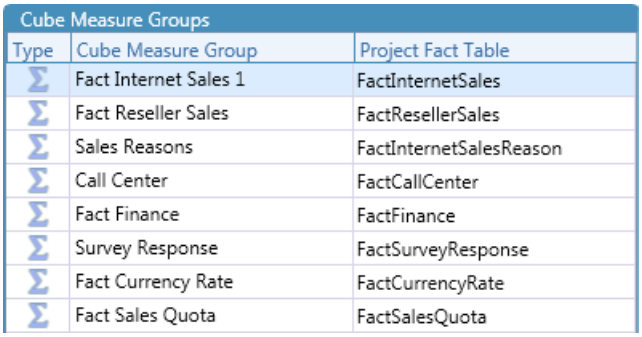
BUTTON	DESCRIPTION
Column	Add a column to the file format.

Cube Editor

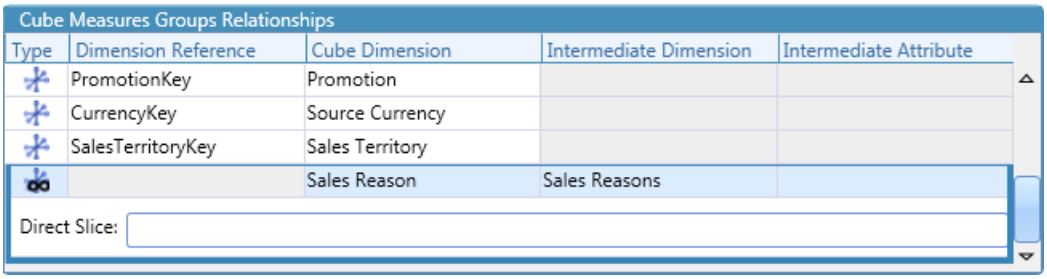
Use the cube editor to create and modify cube dimension and measure groups, along with dimension relationships between cube dimensions and measure groups.



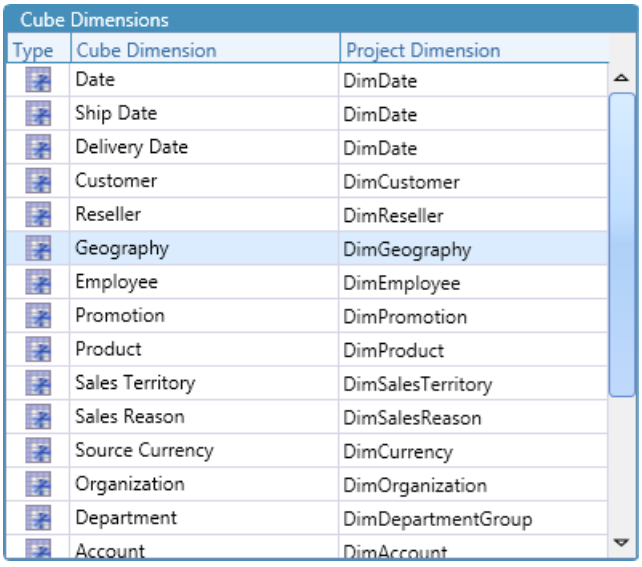
Input a measure group's name and associated fact table in the measure group's data grid row.



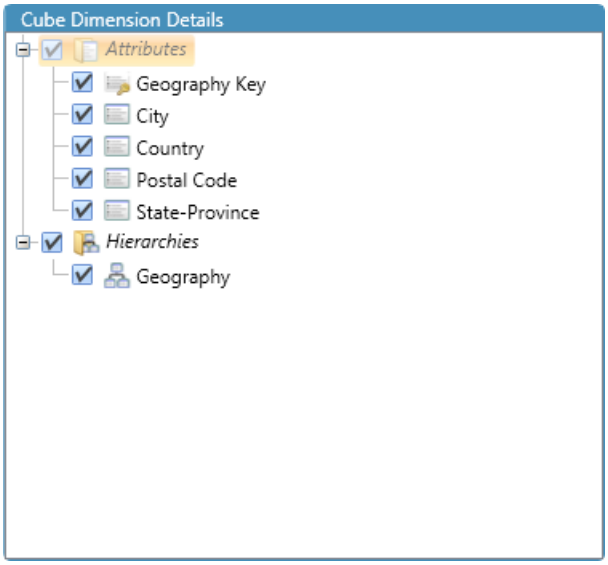
Select a measure group to choose its associated measures, and view and edit it's dimension relationships. Enter a dimension relationship's properties directly in its data grid row.



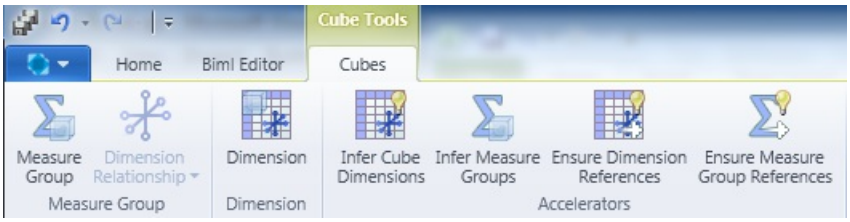
Enter a cube dimension's name and associated project dimension in the cube dimension's data grid row.



Select a cube dimension to view and change its associated cube attributes and hierarchies.



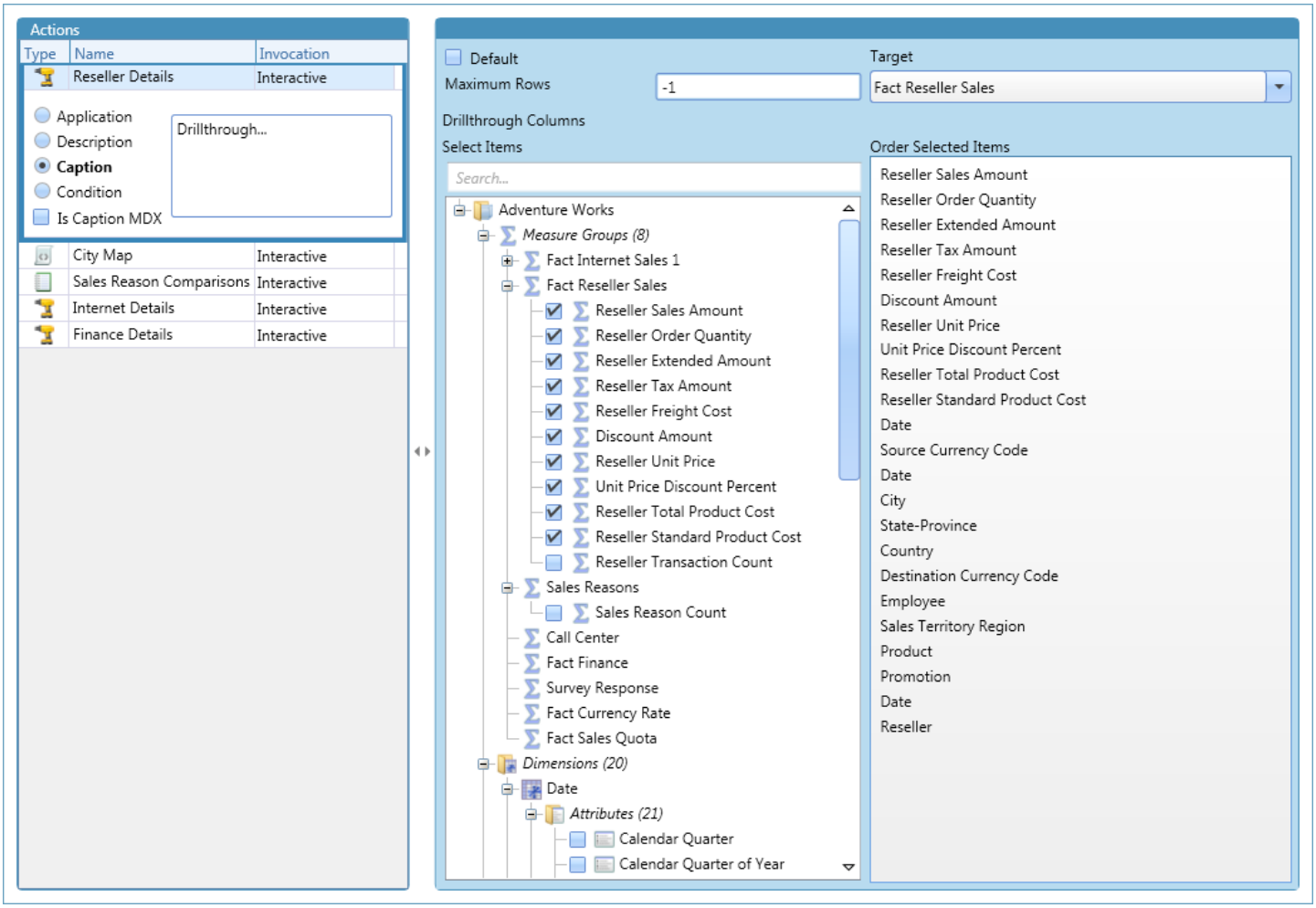
The cube editor's ribbon provides the following functions:



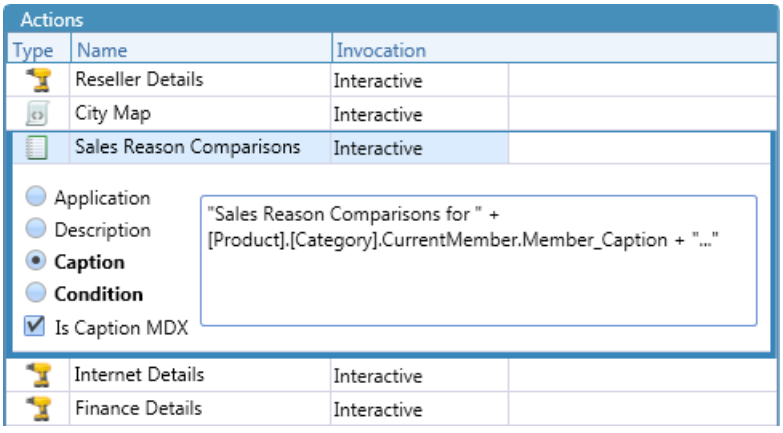
BUTTON	DESCRIPTION
Measure Group	Add a measure group to the cube.
Dimension Relationship	Choose and add a dimension, dimension reference, or many to many relationship to a selected measure group.
Dimension	Add a dimension to the cube.
Infer Cube Dimensions	Generate cube dimensions and dimension relationships from project dimensions and measure groups.
Infer Measure Groups	Generate measure groups from project fact tables.
Ensure Dimension References	In each cube dimension, add references to its associated project dimension's attributes and hierarchies.
Ensure Measure Group References	In each measure group, add references to its associated fact table's measures.

Action Editor

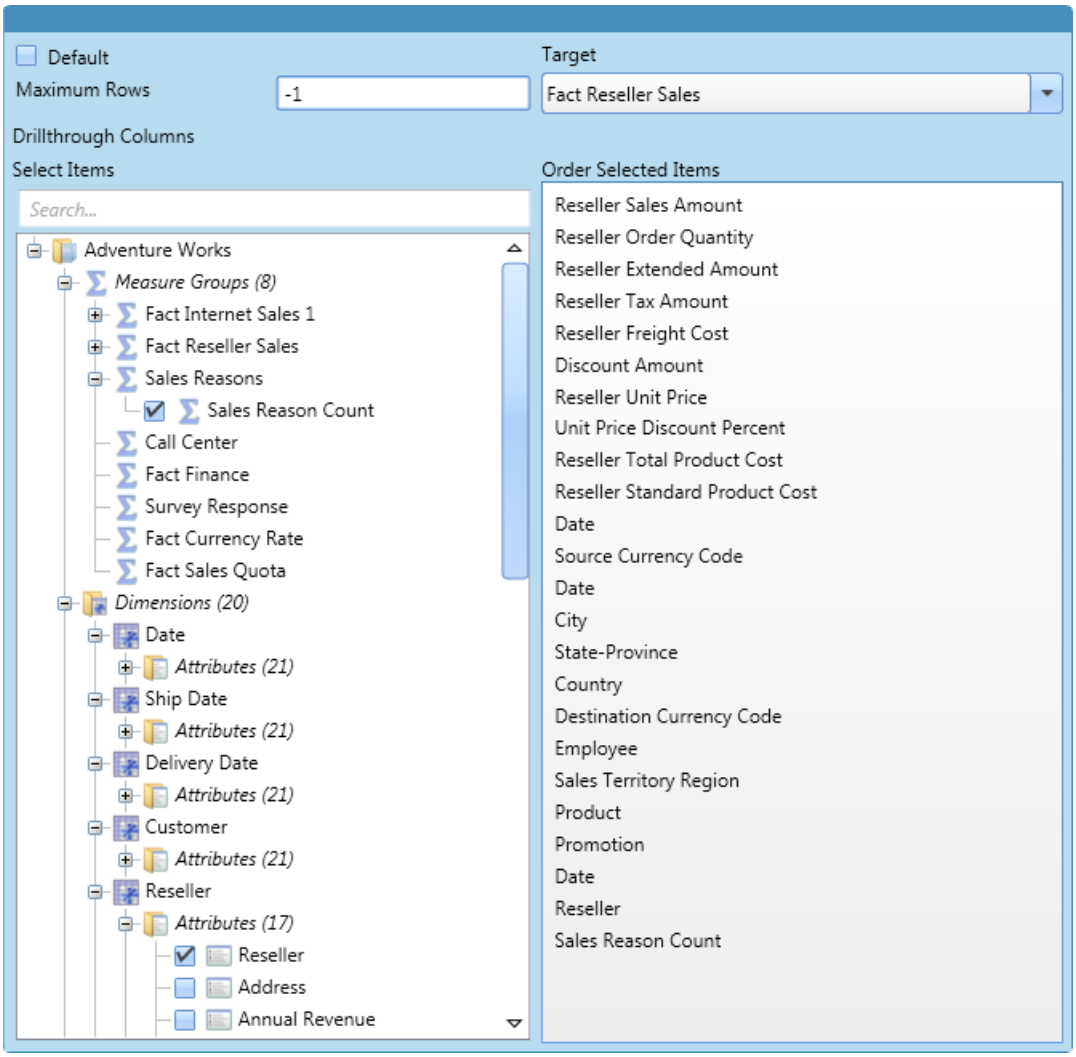
Use the actions editor to view and edit actions for the selected cube.



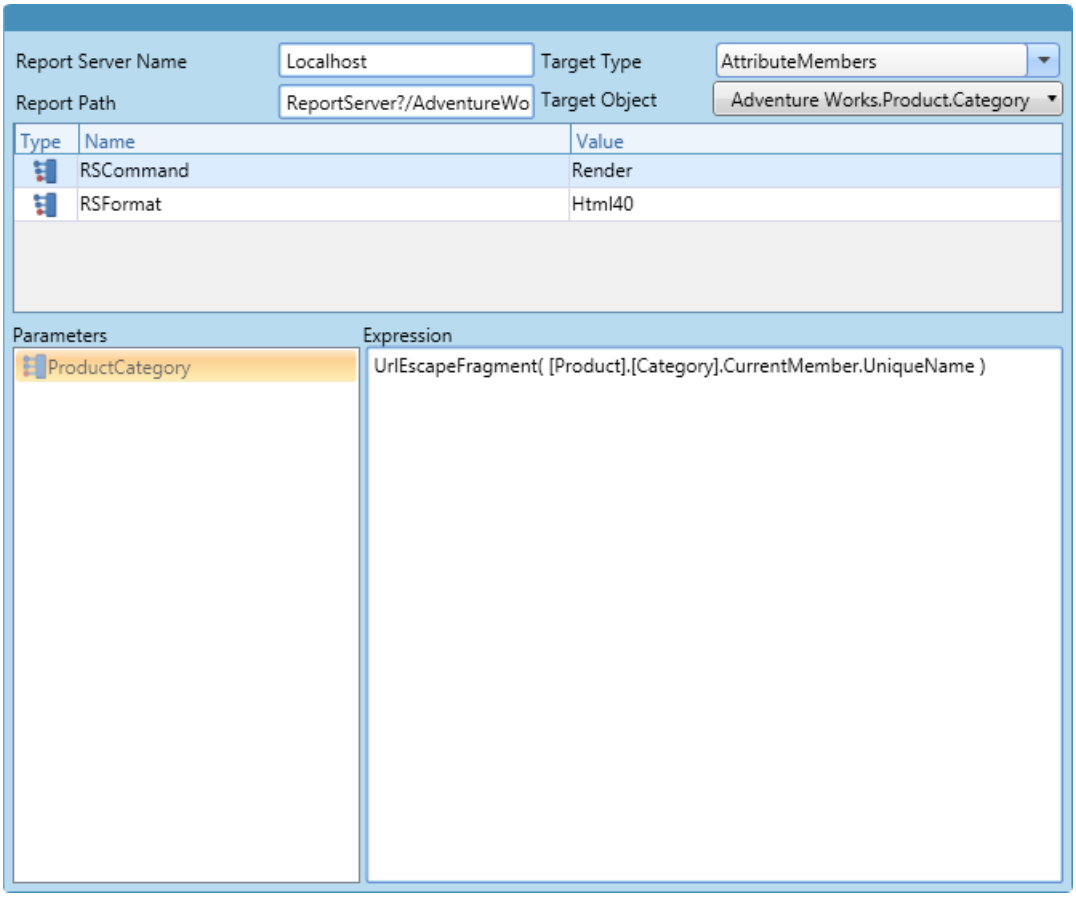
For any action, input its name and invocation type directly in its data grid row. Specify an action's Application, Description, Caption, and Condition in the row's details area.



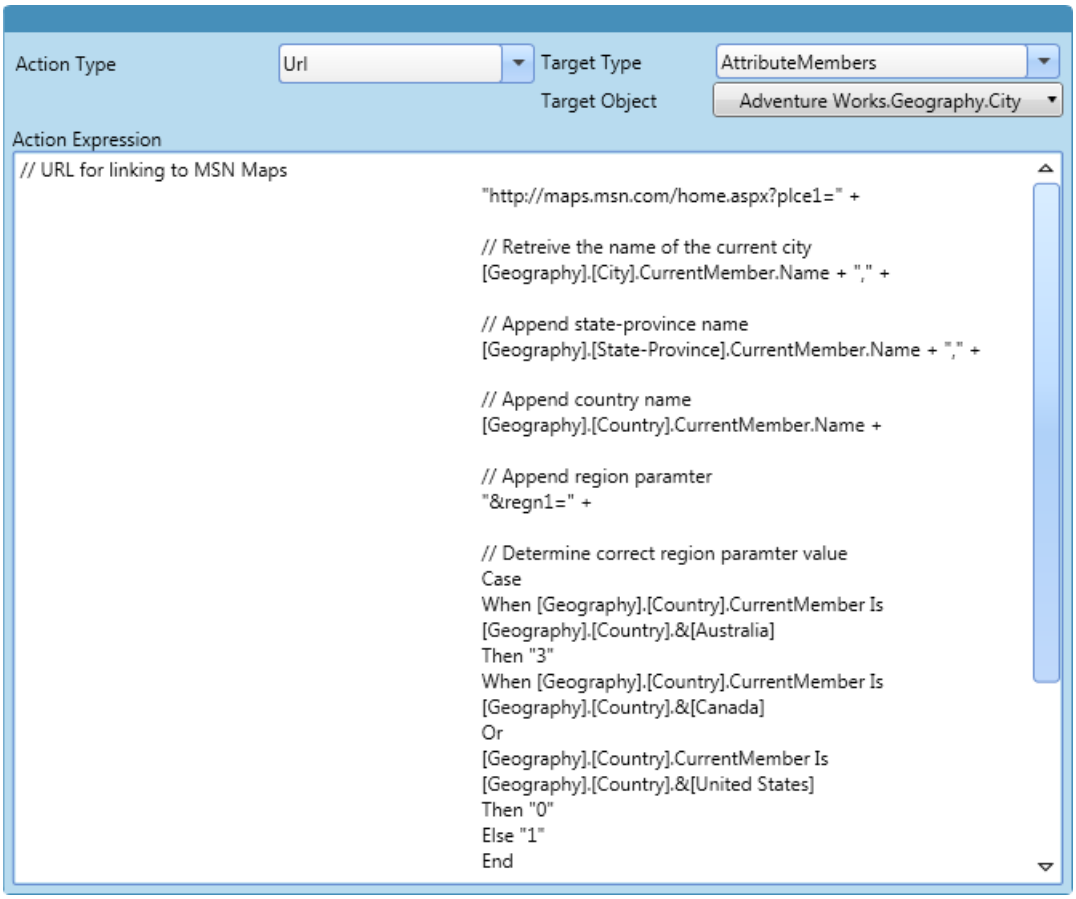
To select drillthrough columns in drillthrough actions, check an attribute or measure to select it. To order the selected items, use drag and drop within the Order Selected Items list box.



For report actions, add a format parameter and supply a value in its data grid row. Additionally, add parameters and enter a MDX expression for each.



For standard actions, supply an MDX expression in the large text editor.

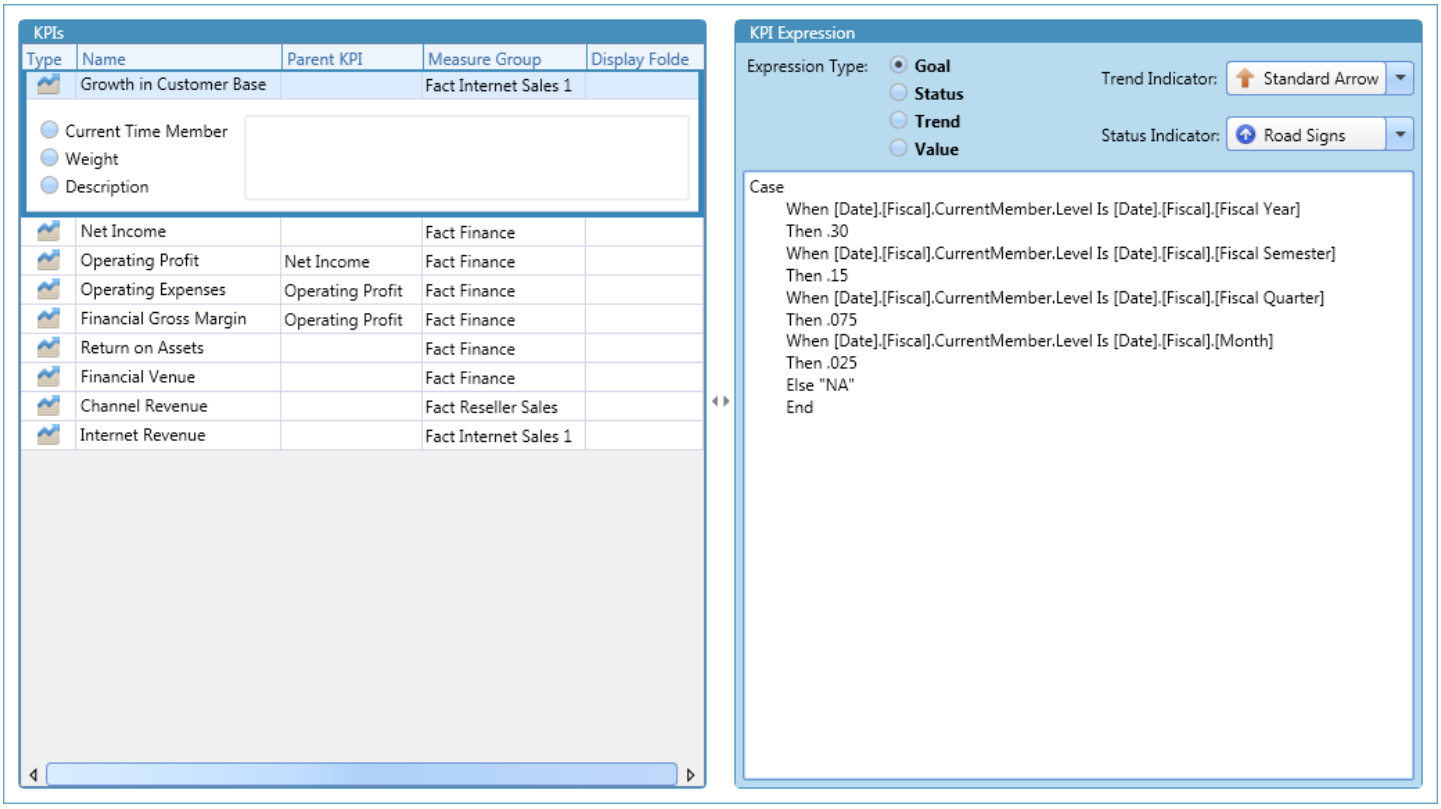


The action editor's ribbon provides the following functions:

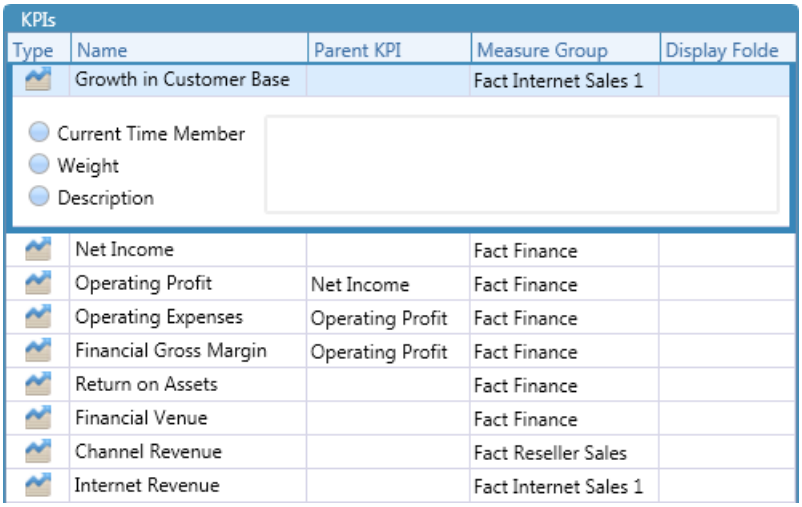
BUTTON	DESCRIPTION
DrillThrough Action	Add a drillthrough action to the cube.
Report Action	Add a report action to the cube.
Standard Action	Add a standard action to the cube.
Format Parameter	Add a format parameter to a selected report action.
Parameter	Add a parameter to a selected report action.

KPI Editor

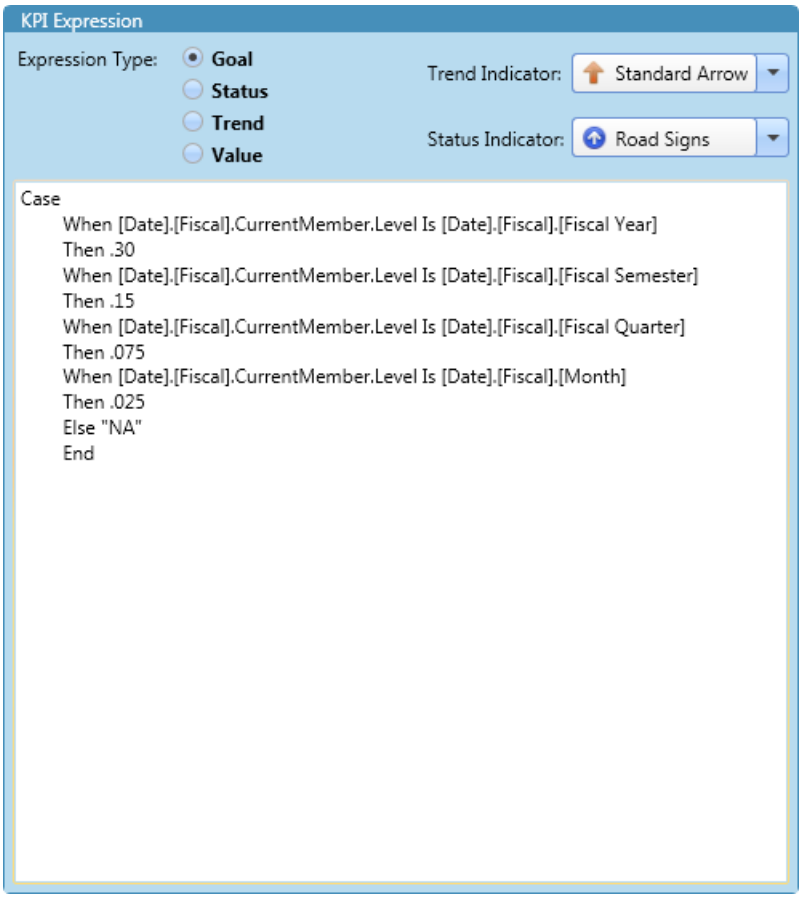
Use the KPI editor to create and edit KPIs.



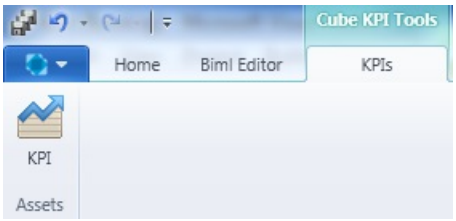
Enter a KPI's name, parent KPI, measure group, and display folder in its data grid row. Specify a KPI's Current Time Member, Weight, and Description in the row's details area.



Select an expression type and type in a MDX expression for that type. Also, specify a trend indicator and status indicator for a KPI.



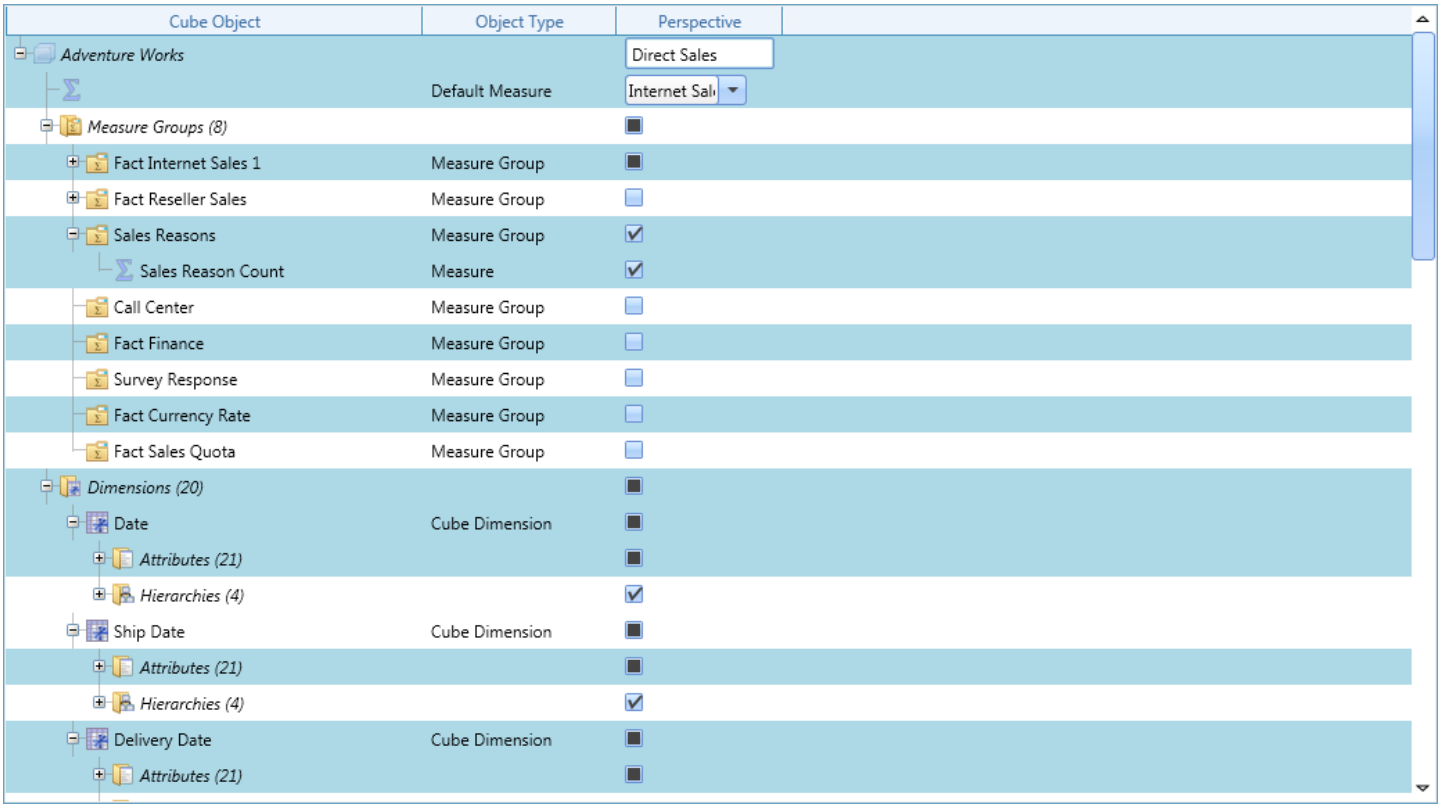
The KPI editor's ribbon provides the following functions:



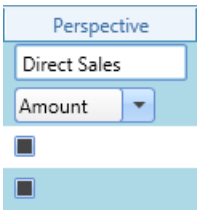
BUTTON	DESCRIPTION
KPI	Add a KPI to the cube.

Perspective Editor

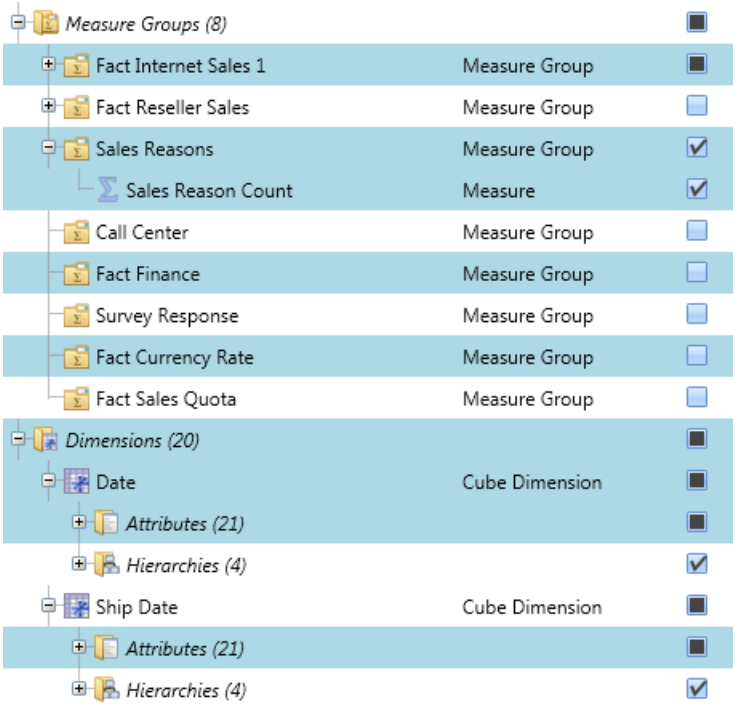
Use the Perspective editor to create and modify perspectives.



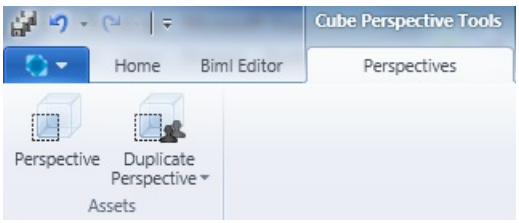
Enter the name of the perspective in its textbox and select its default measure in the Default Measure dropdown.



Select cube objects in the treeview to include them in the perspective.



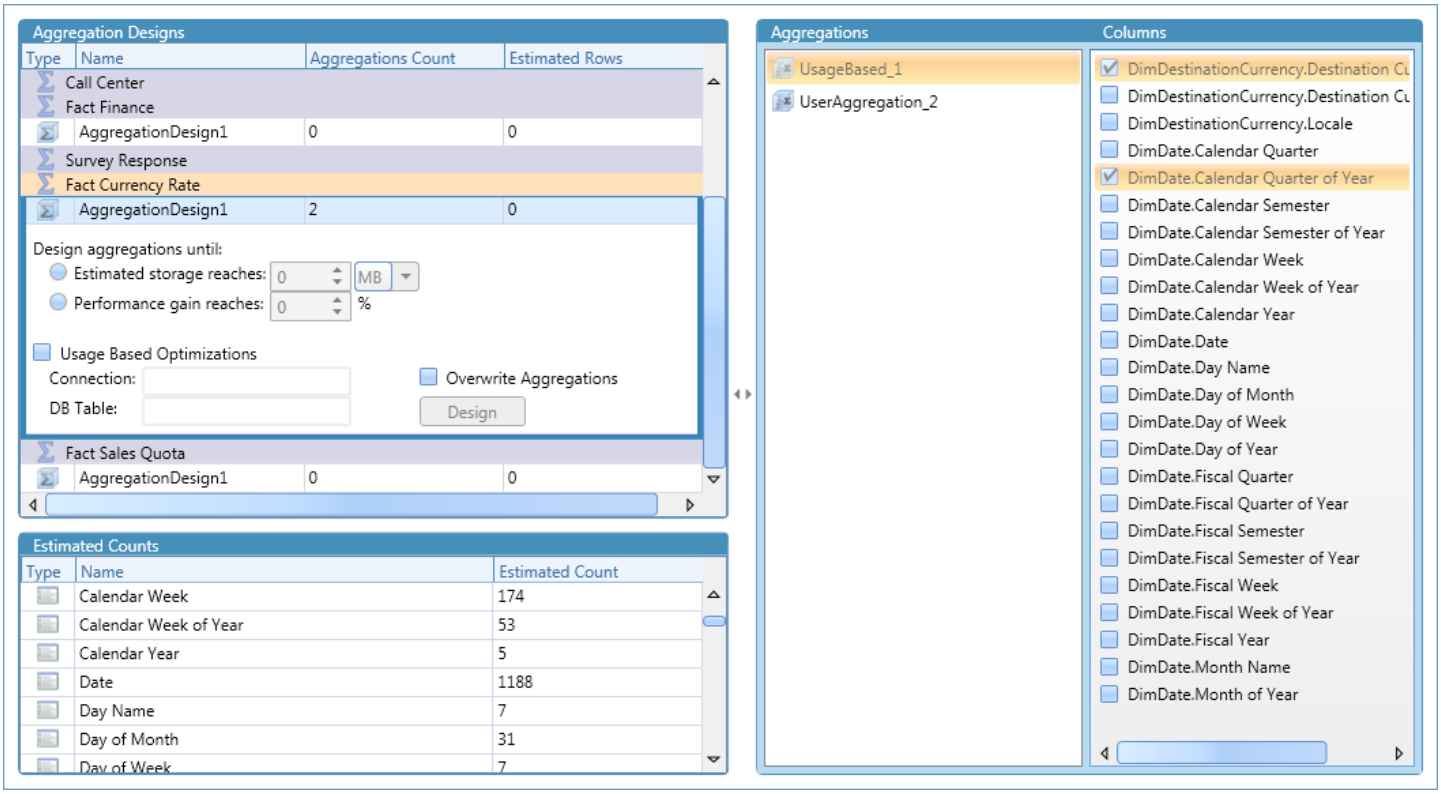
The perspective editor's ribbon provides the following functions:



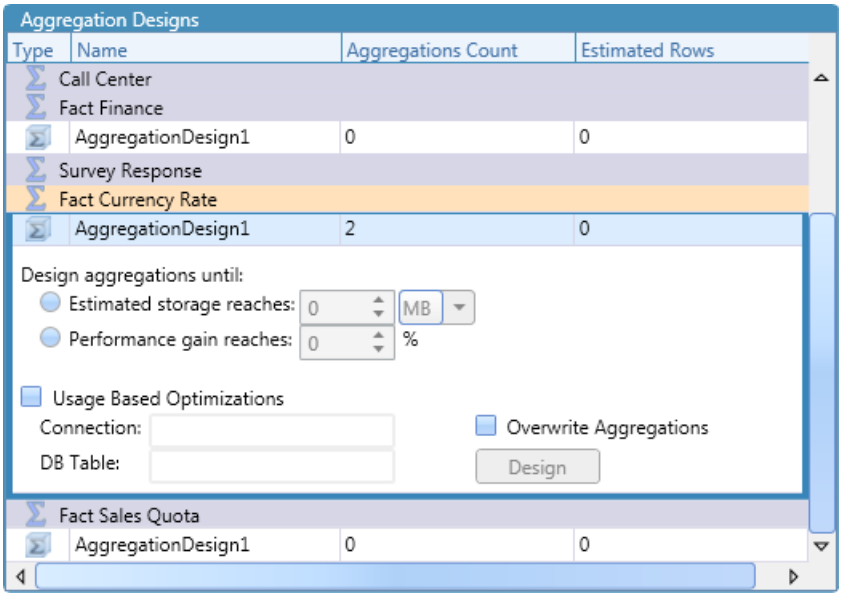
BUTTON	DESCRIPTION
Perspective	Add a perspective to the cube.
Duplicate Perspective	Choose an already created perspective; a copy of that perspective is added to the cube.

Aggregation Design Editor

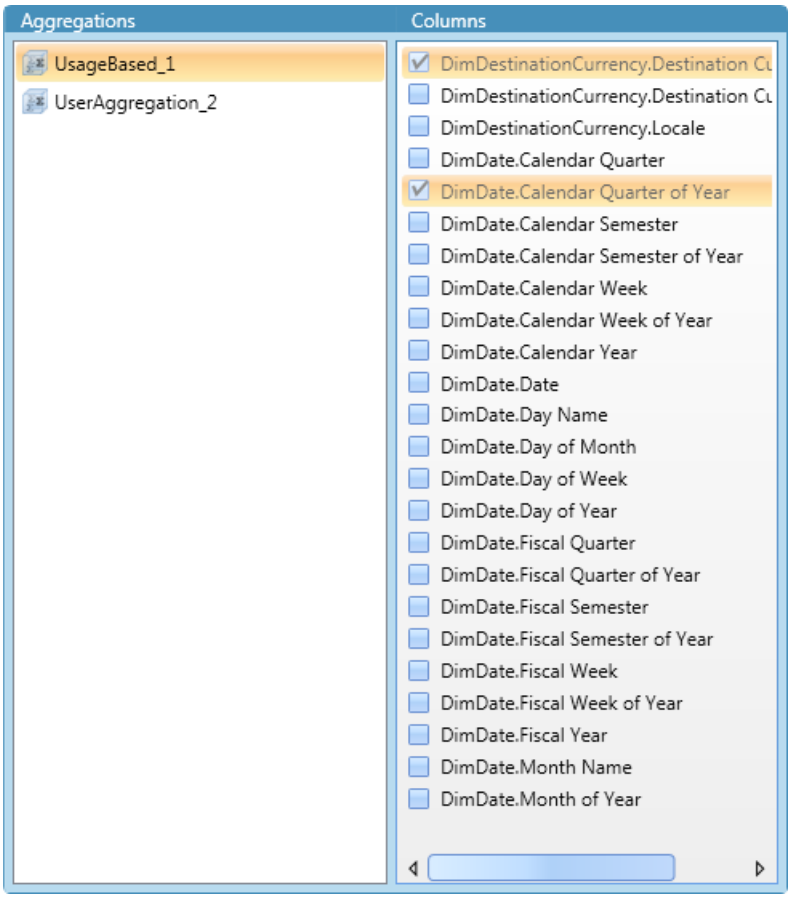
Use the Aggregation Design editor to create and edit aggregation designs, which generate aggregations based on their settings.



Enter an aggregation design's name in its data grid row. Specify the aggregation's design parameters and usage based optimizations in the row's details area. Press Design to generate aggregations.



Select an aggregation to view and change its selected columns.

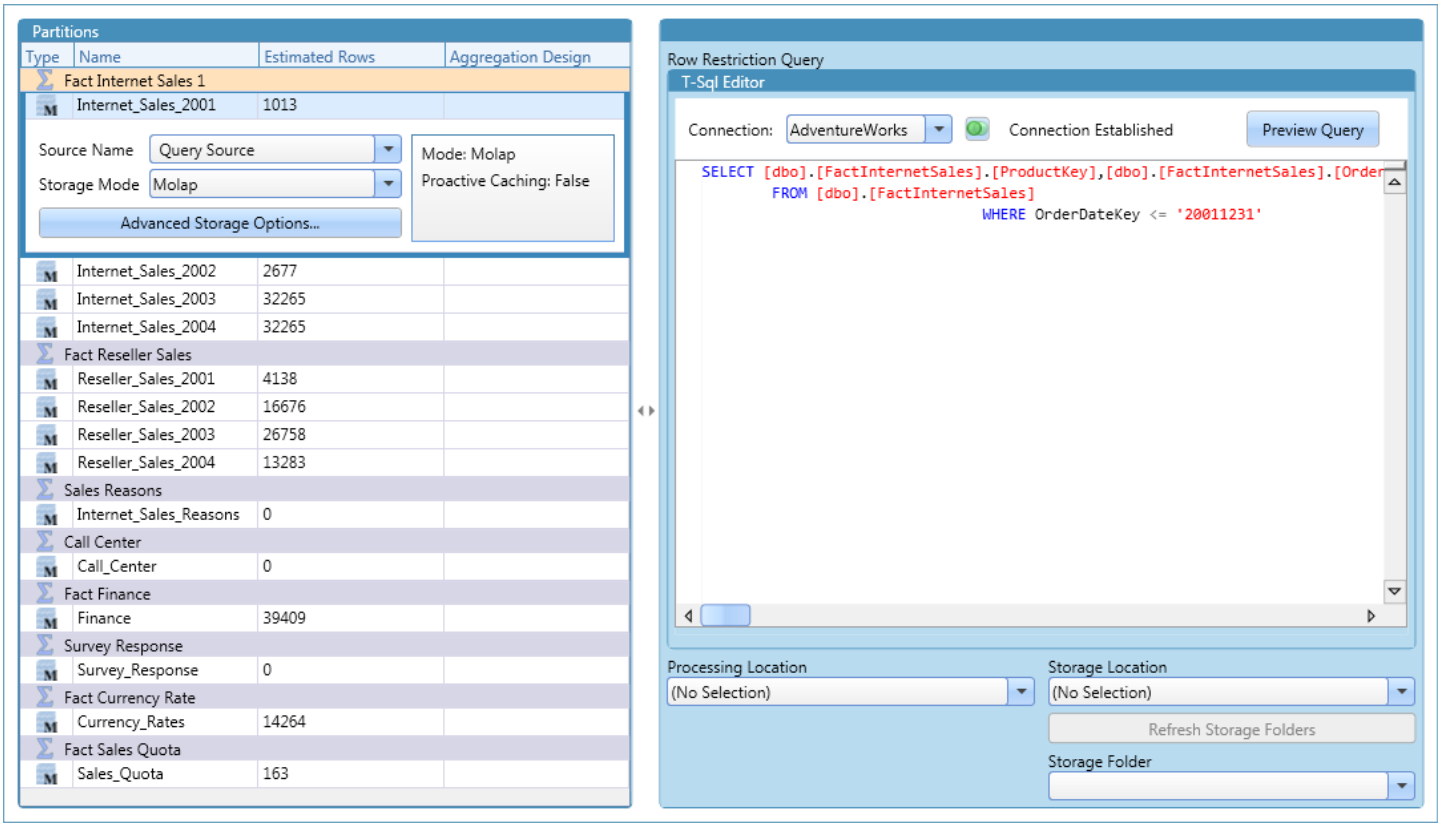


The aggregation editor's ribbon provides the following functions:

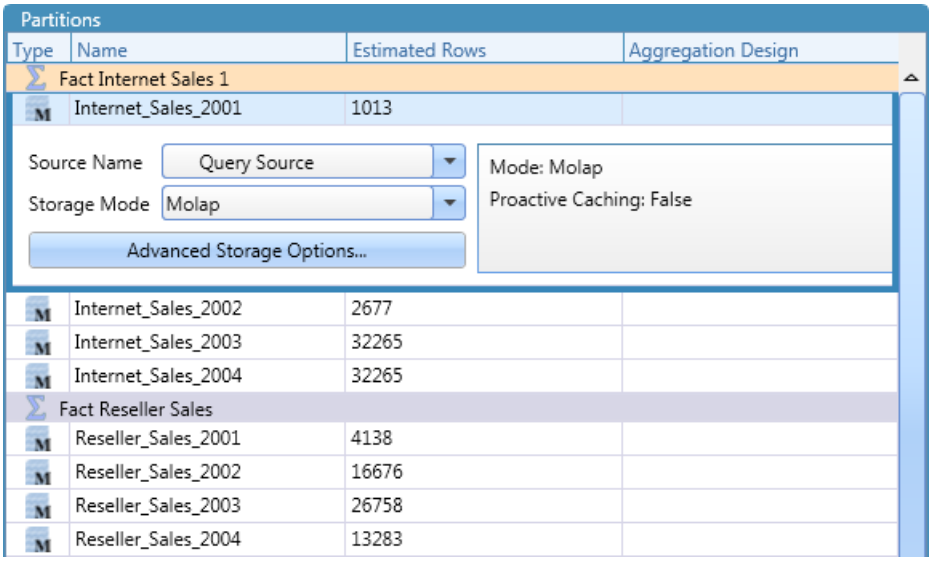
BUTTON	DESCRIPTION
Aggregation Design	Add an aggregation design to the cube.
Aggregation	Add an aggregation to an aggregation design.
Refresh Query Log Properties	Retrieve the QueryLogConnectionString and QueryLogTableName server properties.
Estimate Counts	Generate estimated counts for each measure group and dimension attribute.

Partition Editor

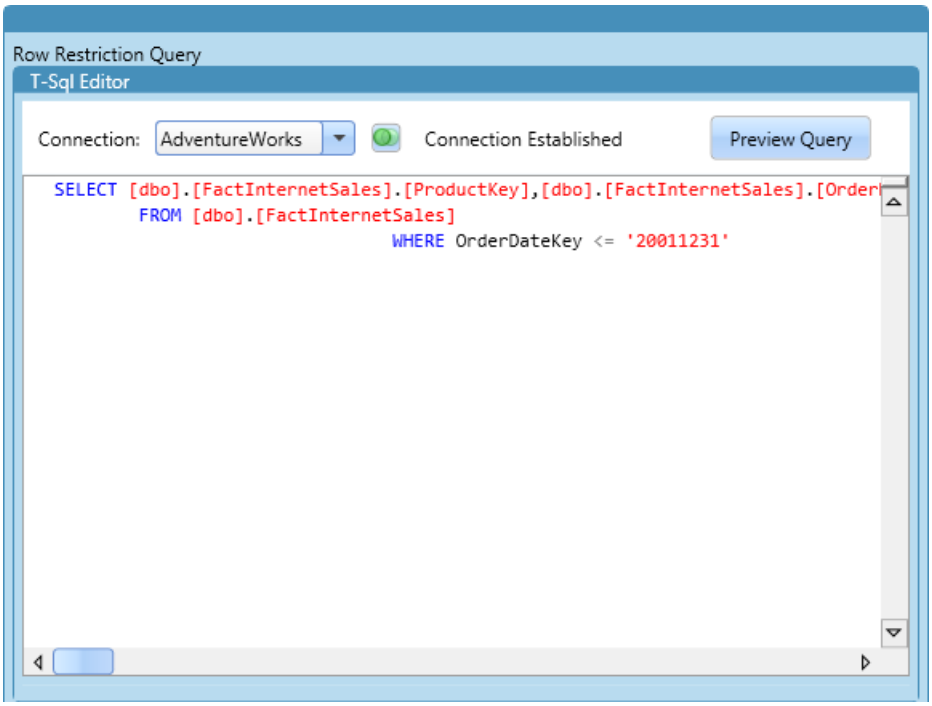
Use the Partitions editor to create and modify partitions, including storage and proactive caching settings, for each cube measure group.



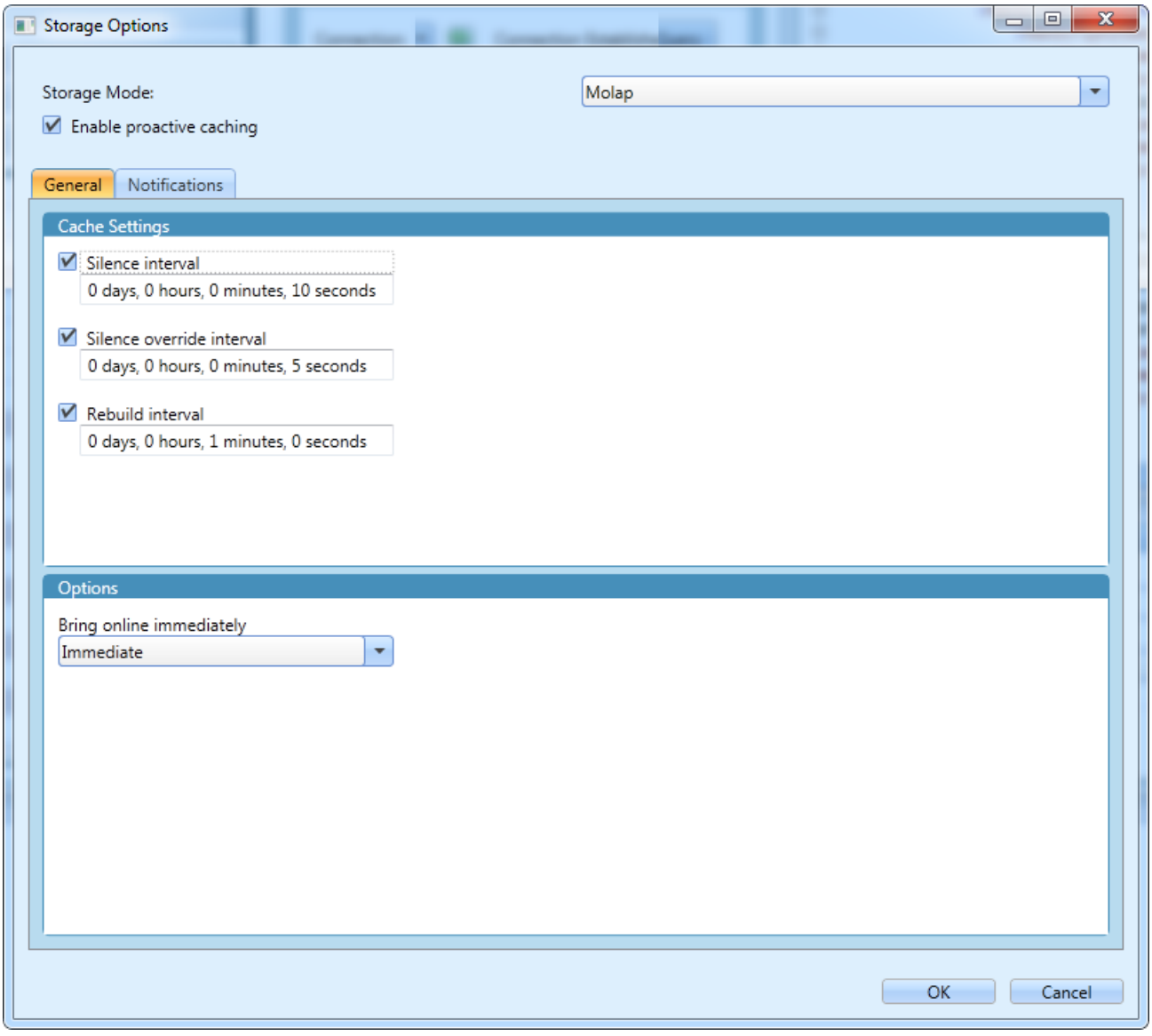
Enter a partition's name, estimated rows count, and associated aggregation design in the partition's data grid row. Select a partition source (table name or query) and storage mode in the row's details area.

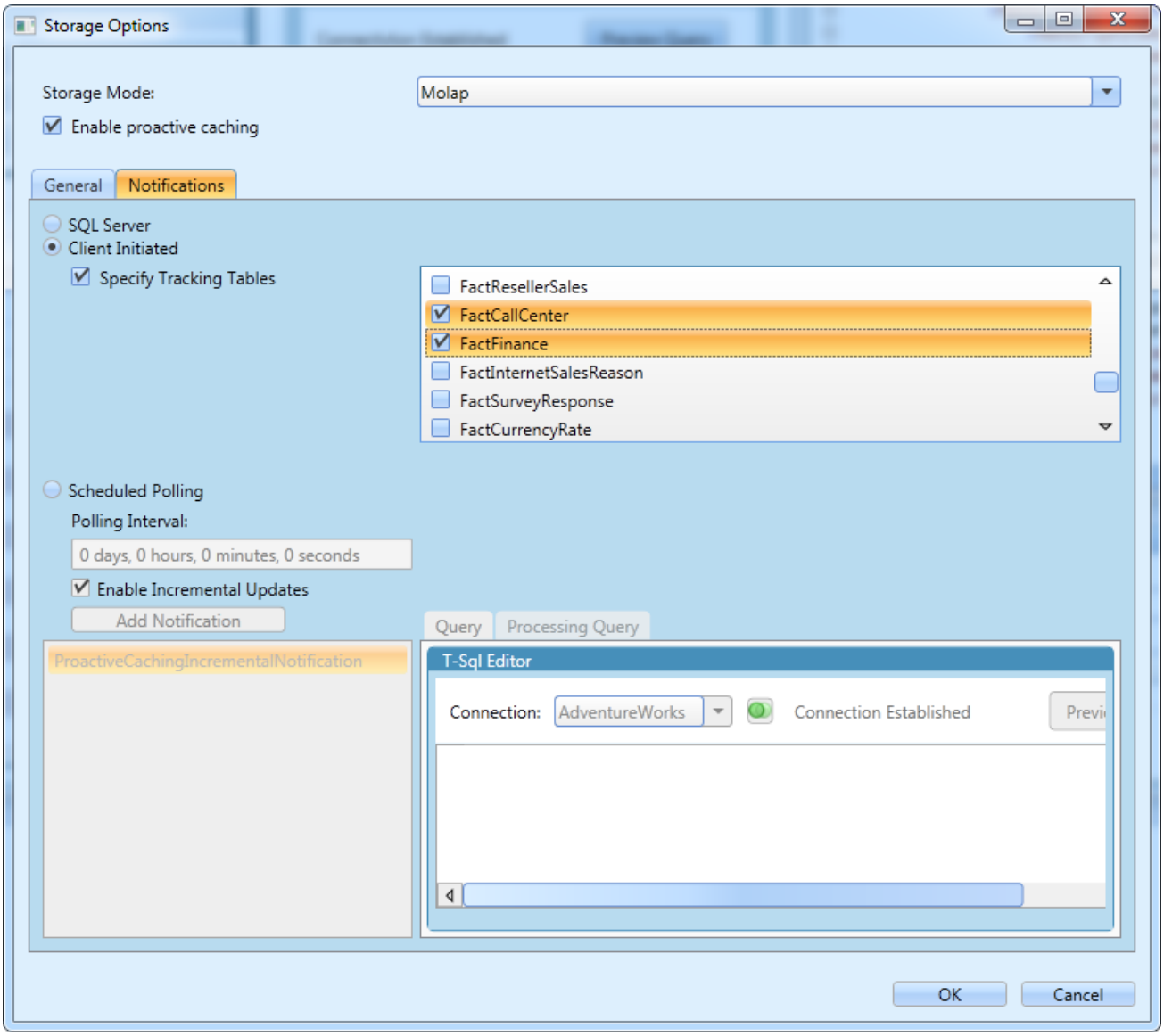


When using a query binding source, enter the query in the T-SQL Editor. Use the Preview Query button to see the query's result.

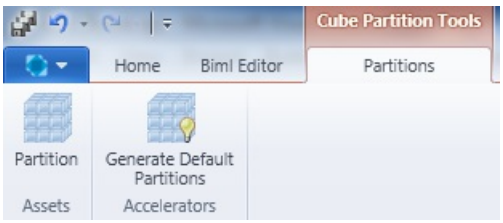


Click on Advanced Storage Options... to open the Storage Options dialog and specify storage mode, proactive caching, cache settings, tracking tables, and incremental updates for the selected partition.





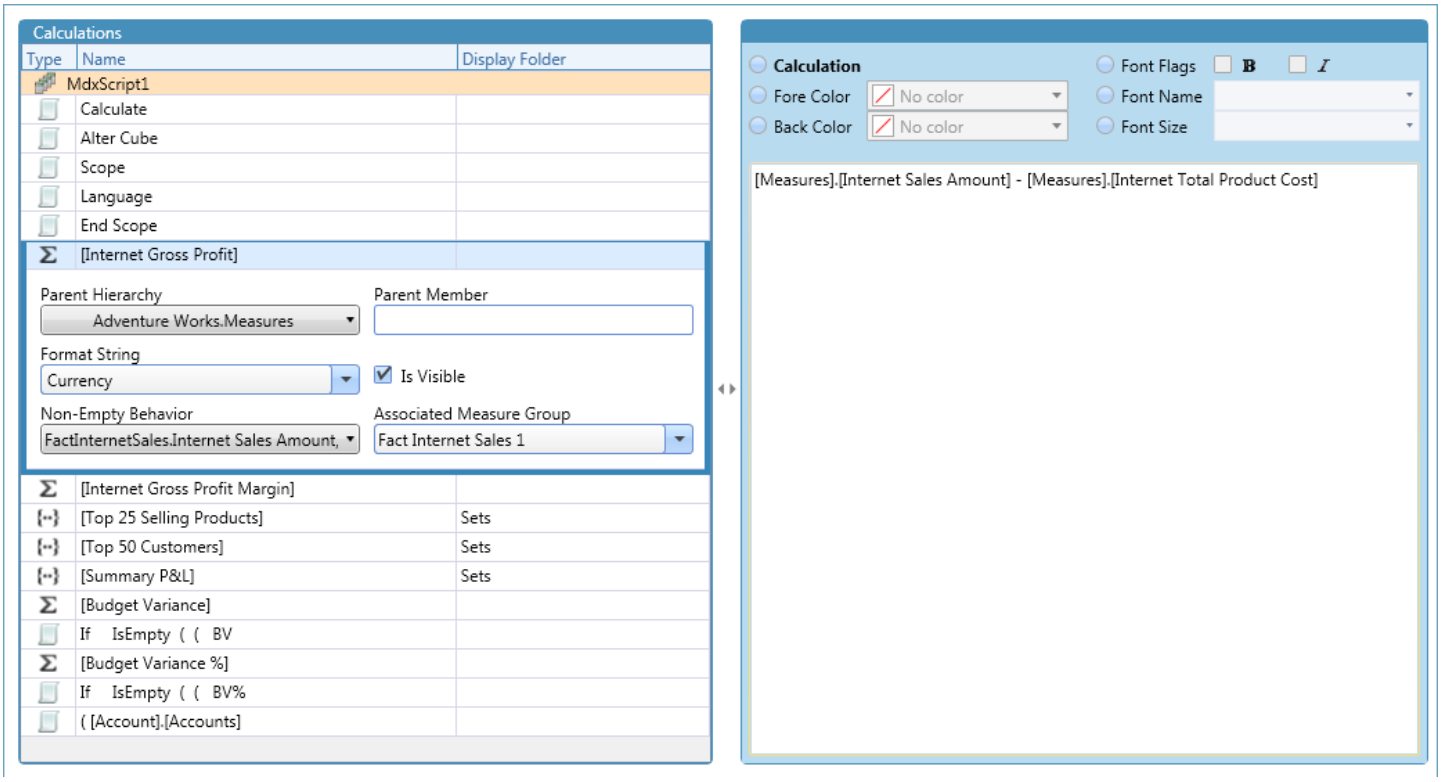
The partition editor's ribbon provides the following functions:



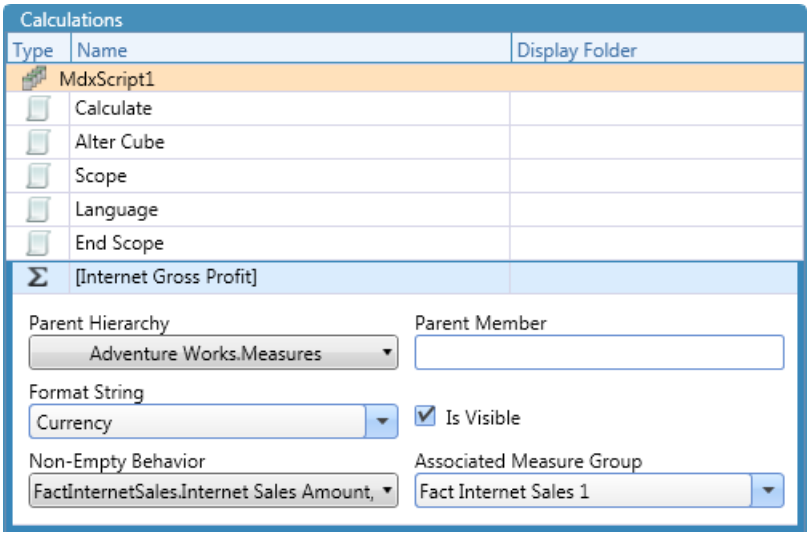
BUTTON	DESCRIPTION
Partition	Add a partition to a selected measure group.
Generate Default Partitions	Add a partition to each cube measure group lacking a partition.

Calculation Editor

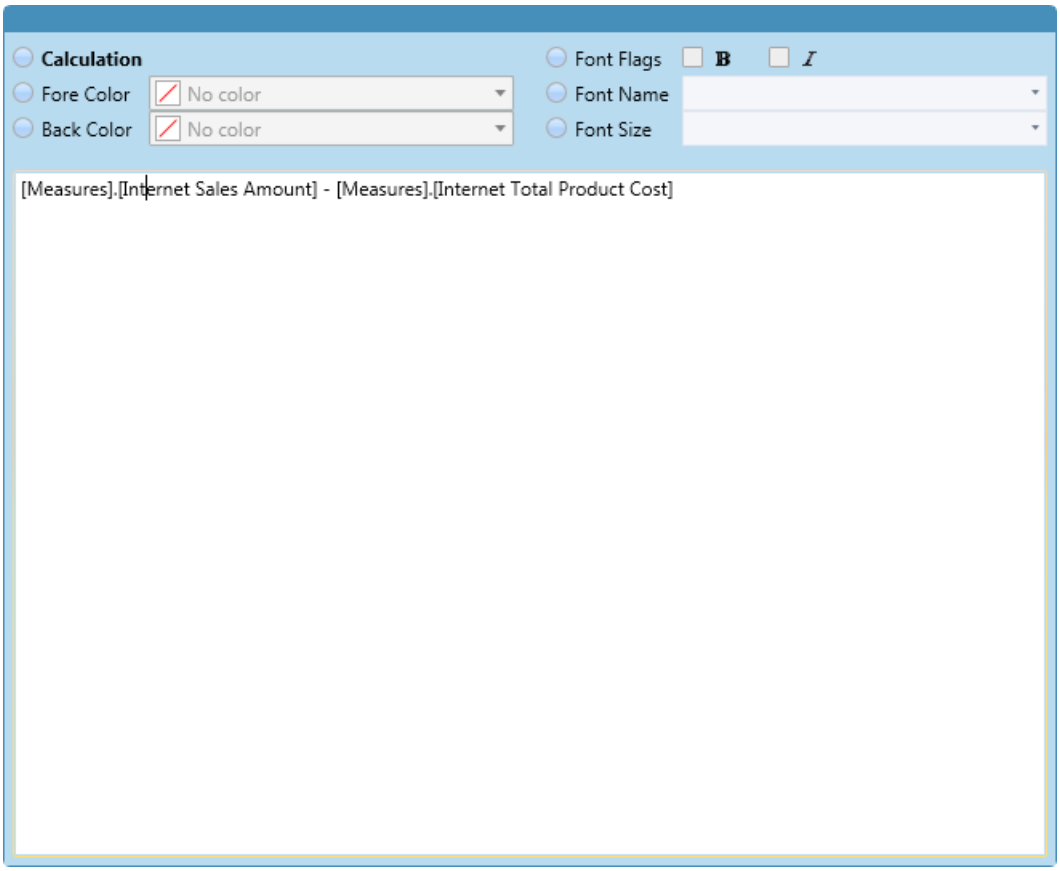
Use the Calculations editor to create and edit calculations, including calculated members, named sets, and MDX script commands.



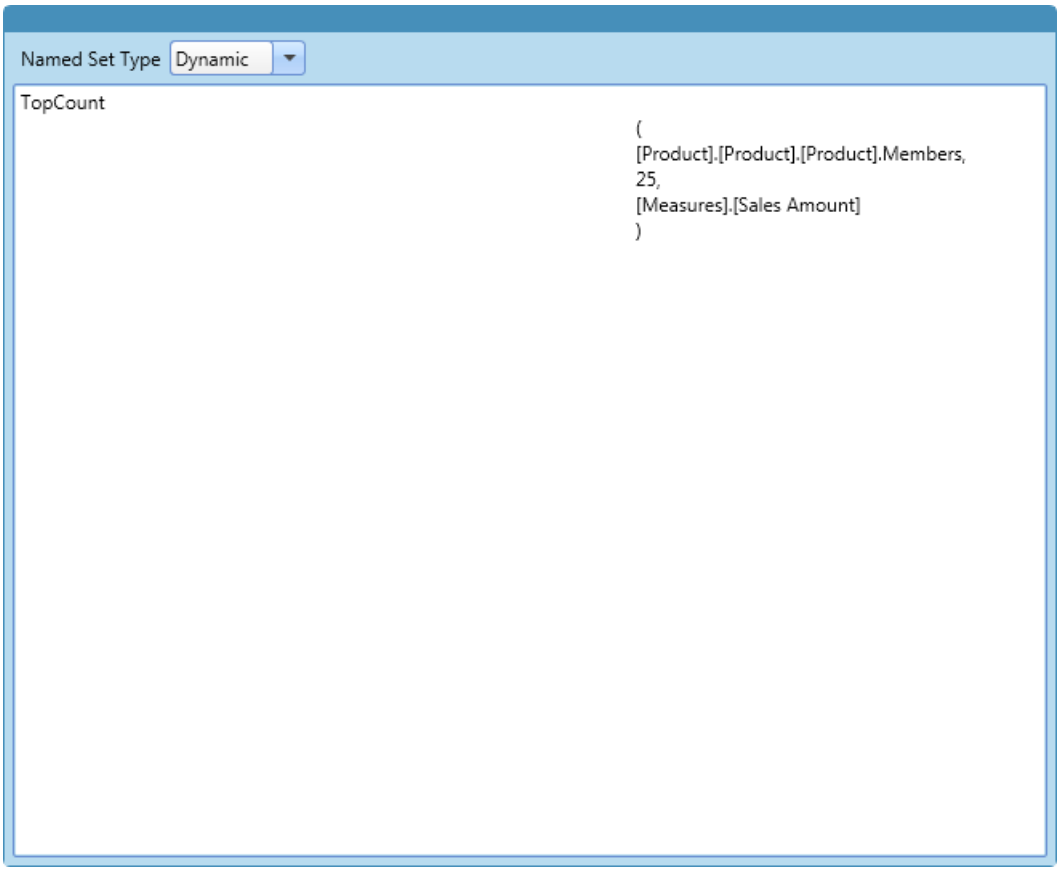
For a Calculated Member, enter its name, parent hiearchy, format string, and other properties in its data grid row.



Select a MDX expression type and then enter a MDX expression below. Alternatively, most types provide a dropdown for entry, generating MDX automatically.



For a Named Set, enter its name in the data grid and enter its MDX expression in the large text editor.

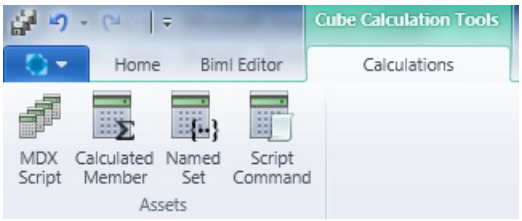


For a Script Command, enter its name in the data grid and enter its MDX expression in the large text editor.

```
// Assignment of format string for empty cells

If
IsEmpty
(
	(
		[Measures].[Amount],
		[Scenario].[Scenario].[Budget]
	)
)
Then
Format_String
(
	[Scenario].[Scenario].[Budget Variance %]
) =
"#;;;Not Budgeted"
End If
```

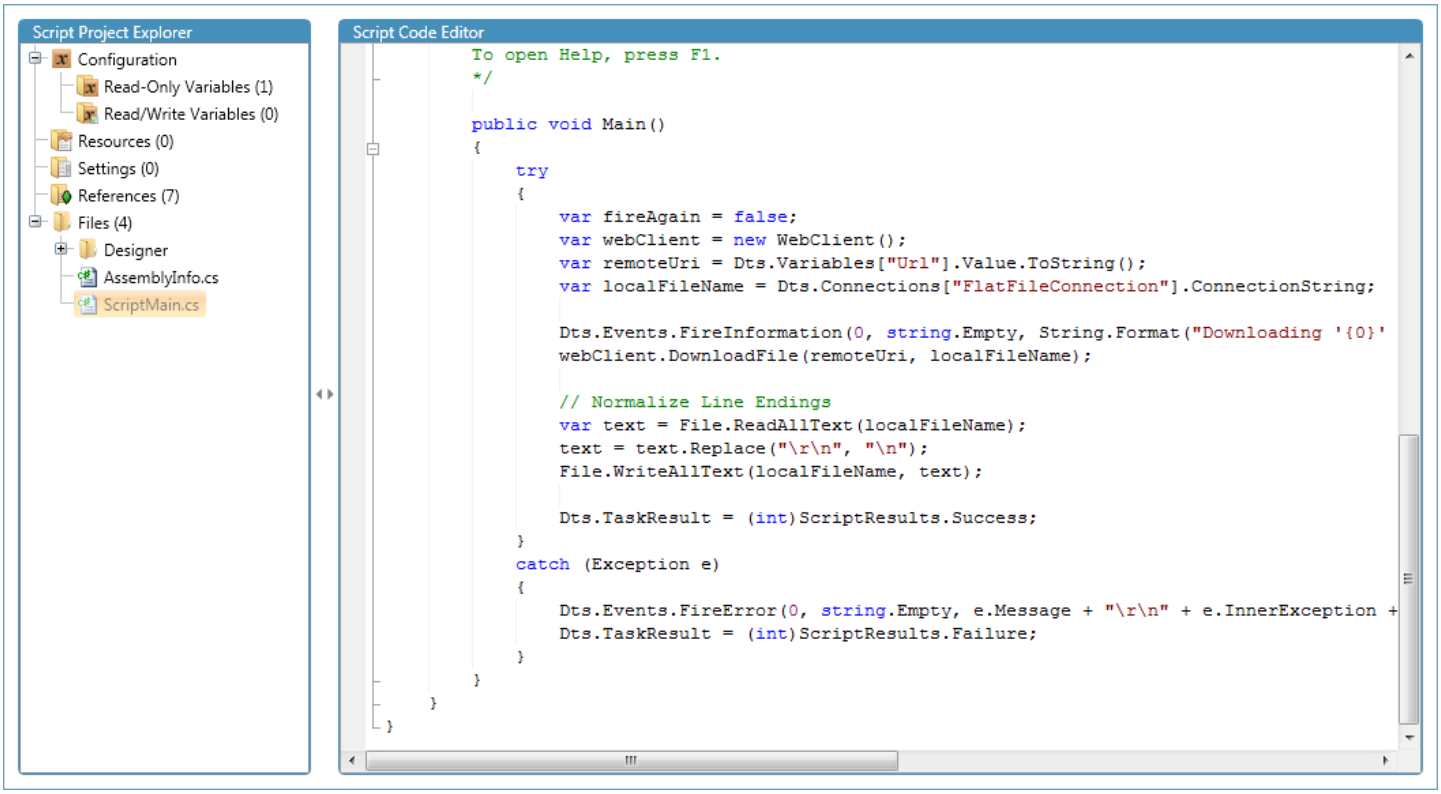
The calculation editor's ribbon provides the following functions:



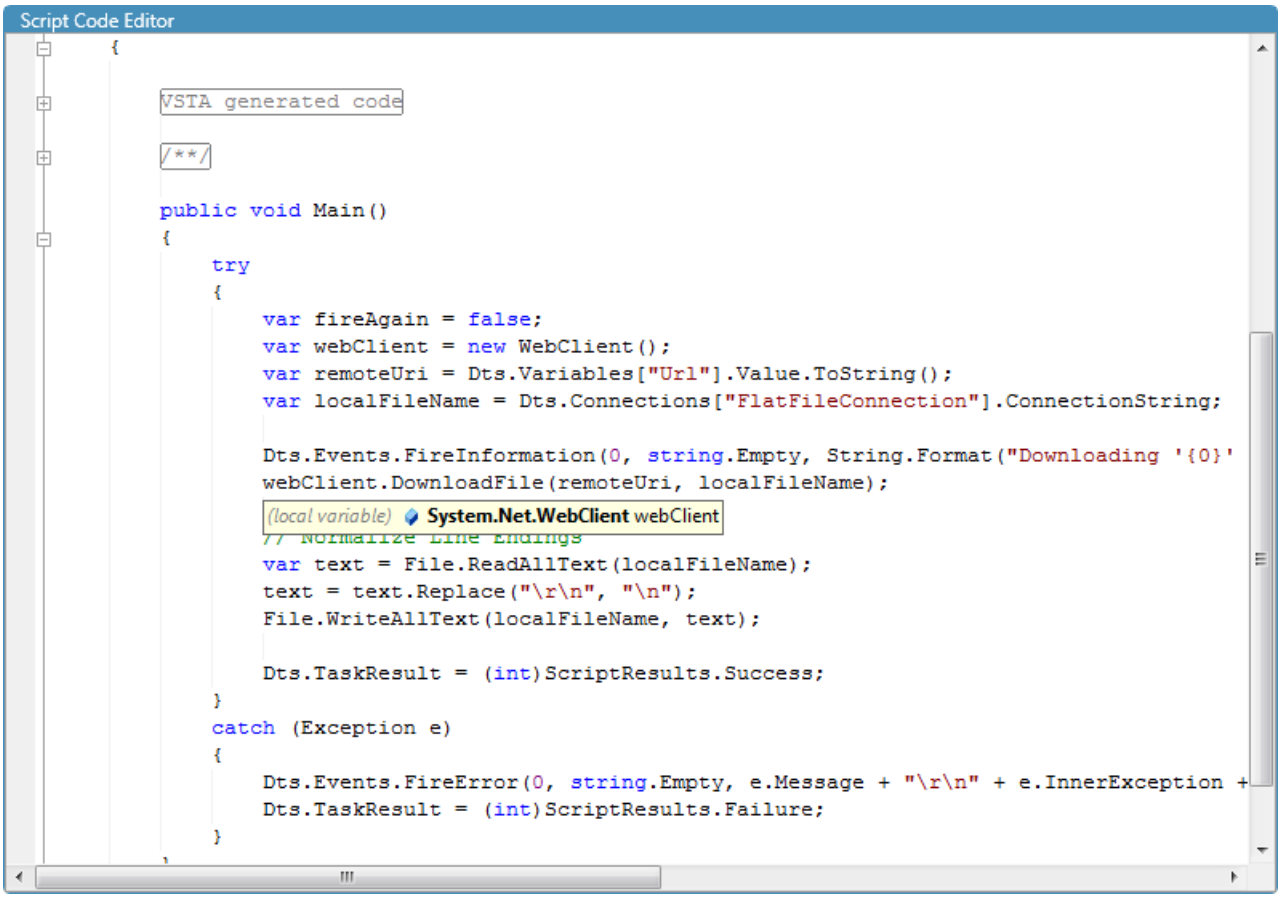
BUTTON	DESCRIPTION
MDX Script	Add a MDX Script to the cube.
Calculated Member	Add a calculated member to a selected MDX Script.
Named Set	Add a named set to a selected MDX Script.
Script Command	Add a script command to a selected MDX Script.

Script Project Editor

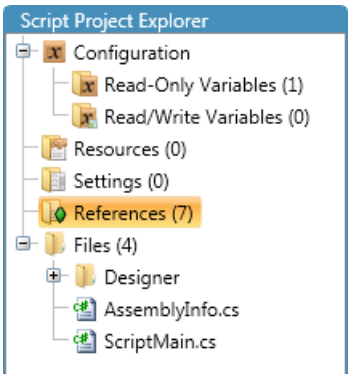
Use the Script Project editor to create and edit script projects that are associated with SSIS script tasks and script components.



Select a file to open it in the editor. The editor provides syntax highlighting, region collapsing, and intelliprompt for efficient coding.



Select other items in the Script Project Explorer (Resources, Settings, References, etc...) to display their editors.

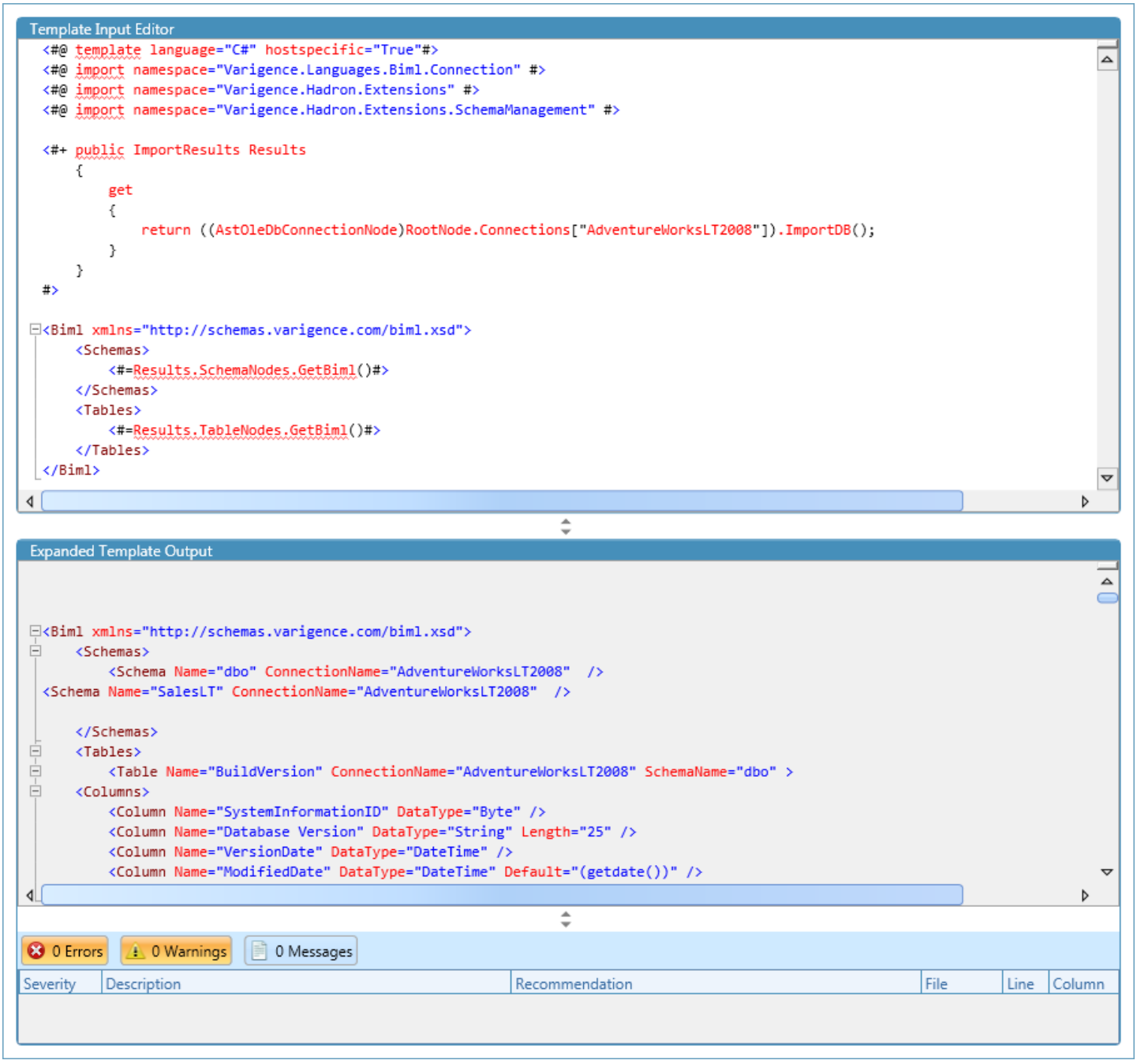


The Script Project editor's ribbon provides the following functions:

BUTTON	DESCRIPTION
File	Add a file to the script project. The default is a language specific file.
Folder	Add a folder to the script project's Files folder.
Resource	Add a resource in the script project.
Setting	Add a setting in the script project.
Reference	Add an assembly reference to the script project.
Read-Only Variable	Add a read-only variable to the script project.
Read/Write Variable	Add a read/write variable to the script project.
Connection Reference	Add a reference, to a project connection, to the script component project.
Output Buffer	Add an output buffer to the script component project.
Column	Add a column to a script component project's input or output buffer.
Test Build	Attempt to compile the script project.

BimlScript Editor

Use the BimlScript editor to create, edit, preview, and execute BimlScript files.



Type Biml and BimlScript into the BimlScript Input Editor.

Template Input Editor

```
<#@ template language="C#" hostspecific="True"#>
<#@ import namespace="Varigence.Languages.Biml.Connection" #>
<#@ import namespace="Varigence.Hadron.Extensions" #>
<#@ import namespace="Varigence.Hadron.Extensions.SchemaManagement" #>

<#+ public ImportResults Results
    {
        get
        {
            return ((AstOleDbConnectionNode)RootNode.Connections["AdventureWorksLT2008"]).ImportDB();
        }
    }
#>

<Biml xmlns="http://schemas.varigence.com/biml.xsd">
    <Schemas>
        <#=Results.SchemaNodes.GetBiml()#>
    </Schemas>
    <Tables>
        <#=Results.TableNodes.GetBiml()#>
    </Tables>
</Biml>
```

As input is entered, the Expanded BimlScript editor displays a live preview of the generated Biml from the BimlScript.

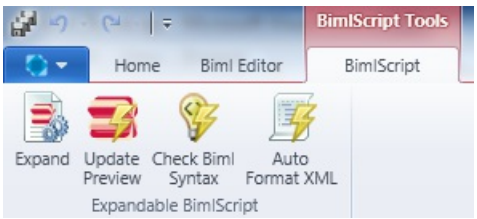
Expanded Template Output

```
<Biml xmlns="http://schemas.varigence.com/biml.xsd">
    <Schemas>
        <Schema Name="dbo" ConnectionName="AdventureWorksLT2008"  />
 <Schema Name="SalesLT" ConnectionName="AdventureWorksLT2008"  />

    </Schemas>
    <Tables>
        <Table Name="BuildVersion" ConnectionName="AdventureWorksLT2008" SchemaName="dbo" >
    <Columns>
        <Column Name="SystemInformationID" DataType="Byte" />
        <Column Name="Database Version" DataType="String" Length="25" />
        <Column Name="VersionDate" DataType="DateTime" />
        <Column Name="ModifiedDate" DataType="DateTime" Default="(getdate())" />
```

Once finished, press the Expand ribbon button to generate real Biml from the BimlScript that is added to the project.

The BimlScript editor's ribbon provides the following functions:



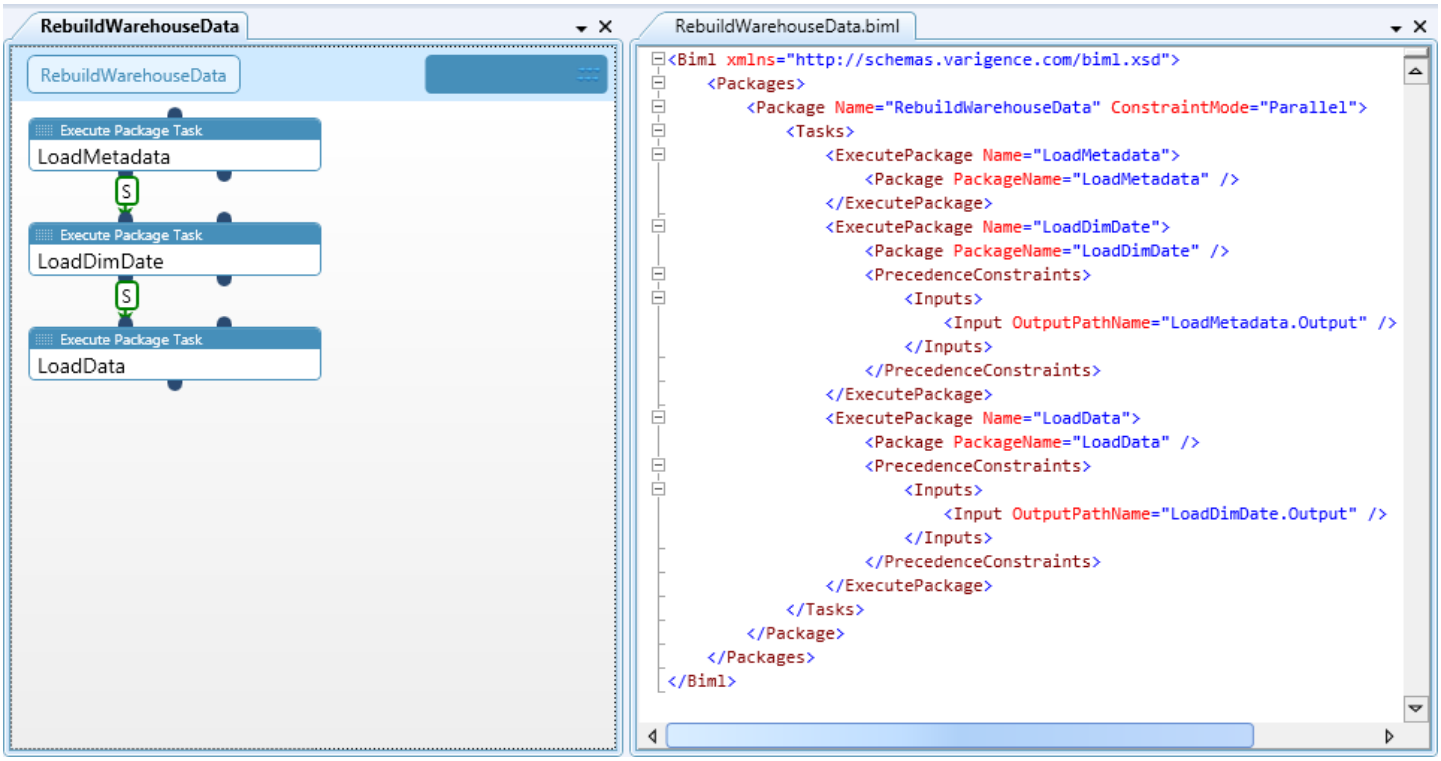
BUTTON	DESCRIPTION
Expand	Execute the BimlScript; the produced Biml is applied to the project.
Update Preview	Refresh the Expanded BimlScript Output editor
Check Biml Syntax	Evaluate that correct Biml syntax is used in the BimlScript output.
Auto Format XML	Activate to have BimlScript output be formatted automatically.

Biml Editor

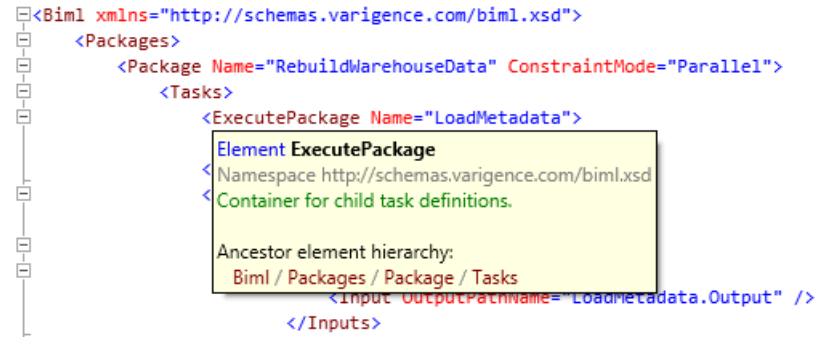
Use the Biml editor to edit Biml files.

```
<Biml xmlns="http://schemas.varigence.com/biml.xsd">
    <Packages>
        <Package Name="RebuildWarehouseData" ConstraintMode="Parallel">
            <Tasks>
                <ExecutePackage Name="LoadMetadata">
                    <Package PackageName="LoadMetadata" />
                </ExecutePackage>
                <ExecutePackage Name="LoadDimDate">
                    <Package PackageName="LoadDimDate" />
                    <PrecedenceConstraints>
                        <Inputs>
                            <Input OutputPathName="LoadMetadata.Output" />
                        </Inputs>
                    </PrecedenceConstraints>
                </ExecutePackage>
                <ExecutePackage Name="LoadData">
                    <Package PackageName="LoadData" />
                    <PrecedenceConstraints>
                        <Inputs>
                            <Input OutputPathName="LoadDimDate.Output" />
                        </Inputs>
                    </PrecedenceConstraints>
                </ExecutePackage>
            </Tasks>
        </Package>
    </Packages>
</Biml>
```

Every item in the Logical View has an associated Biml file. While the item's editor and the property grid can typically provide complete customization for an item, it may be convenient to alter Biml directly.



Syntax highlighting, quick info popups, and Intelliprompt are all available.



The Biml editor's ribbon provides the following functions:

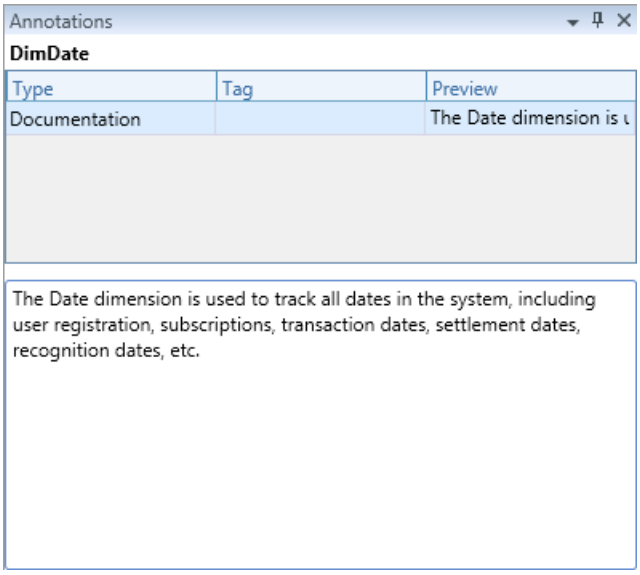
BUTTON	DESCRIPTION
Paste	Paste text from the clipboard into the Biml document.
Cut	Remove selected text from the document and place it on the clipboard.
Copy	Copy selected text to the clipboard.
Font	Choose a font for the Biml editor text. The default is Consolas.
Size	Choose a text size for the Biml editor text.
Def	Returns font related settings to their defaults.
Type	Choose a type of text to change its foreground or background color.
Background	Select the background color for a type of text.
Foreground	Select the foreground color for a type of text.
Find	Opens the Find and Replace tool window.
Show Intellisense	Displays an Intelliprompt popup at the cursor location.
Toggle All	Choose to toggle regions in the Biml document.
Include File	Add a new or existing Biml include file to the open project.
Emittable File	Add a new or existing Biml emittable file to the open project.

BUTTON	DESCRIPTION
Instant Recompile	Immediately recompile the Biml file.
Add	Add a project configuration.
Remove	Remove a project configuration.

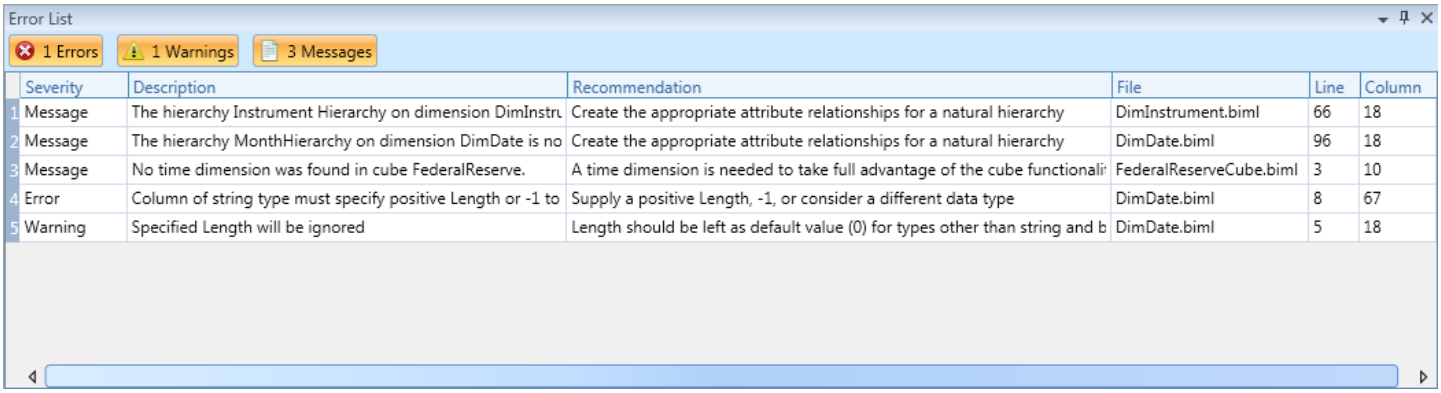
Common Tool Windows Overview

There are several tool windows that provide access to project assets and aid in development. They are listed below in alphabetical order.

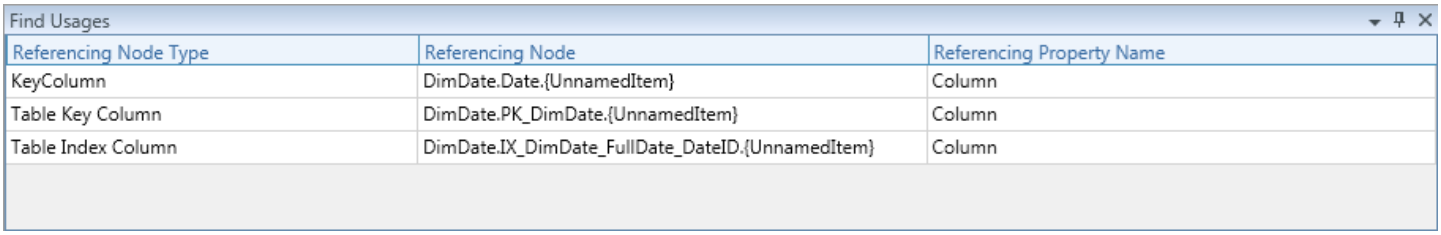
- Annotations - Choose a type and a tag in its data grid row. In the text box beneath the data grid row, enter the annotation text. The preview cell displays the annotation's initial text.



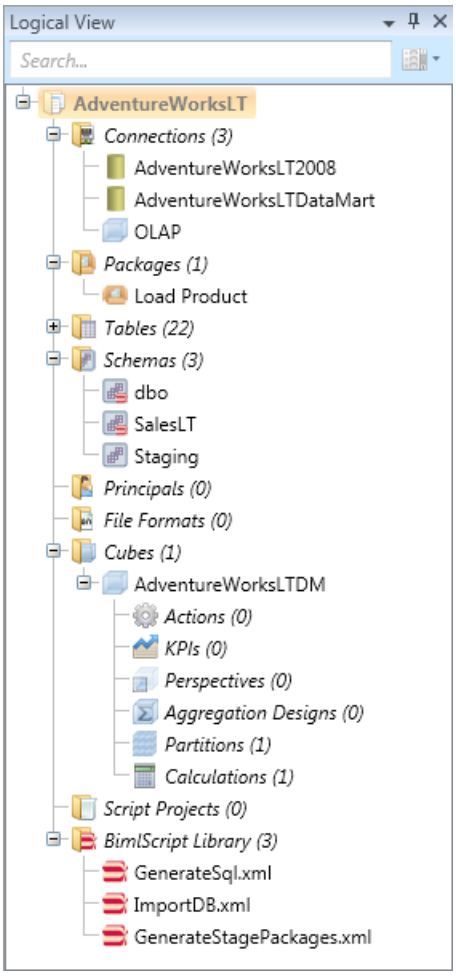
- Error List - Each data grid row contains a validation issue, and provides the issue's severity, an issue description, recommendation for fixing the issue, the file where the issue occurs, and the line and column for the issue. Double clicking on an issue navigates to the issue in the Mist UI.



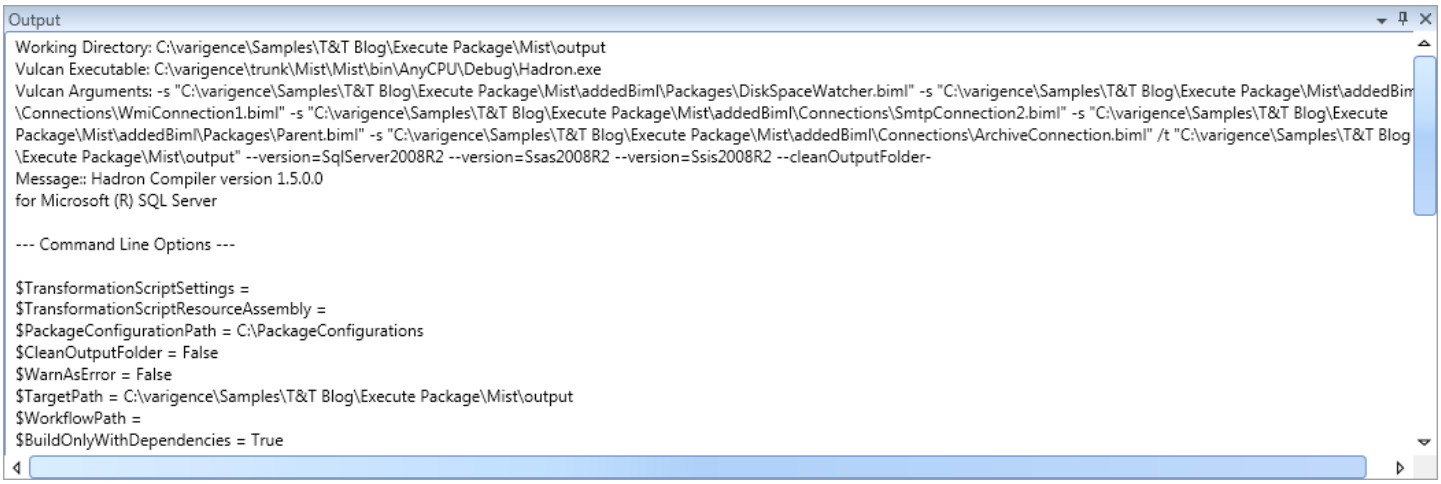
- Find Usages - Each data grid row displays a reference to a selected item the Mist UI. Each row indicates the type of referencing node, the actual node, and the node's referencing property.



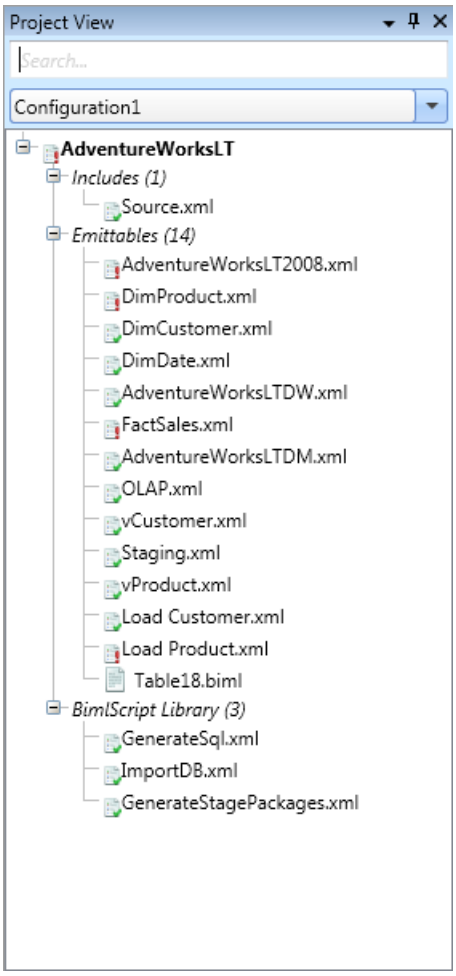
- Logical View - Displays all the items in a project using a hierarchical structure. The root item is the project itself. Type in the search textbox to filter the items based on their displayed names. Right click on groups to add new assets. Right click on items to view, rename, and remove them.



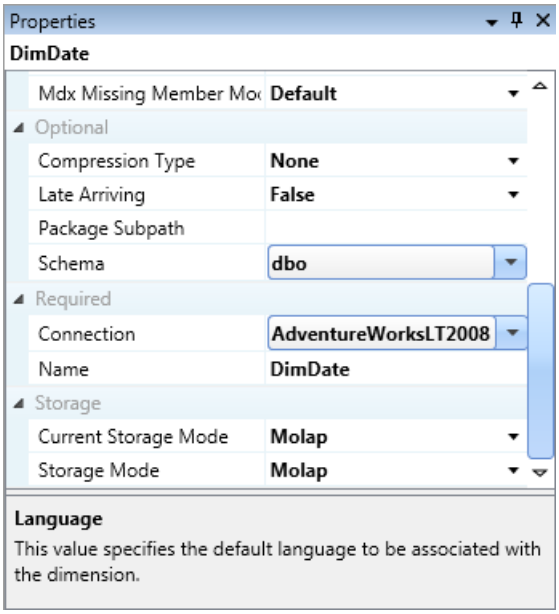
- Output - Displays output when building a project or executing the Build & Run or Build & Open in BIDS commands.



- Project View - Displays all Biml files in a project, grouped by files with assets to be emitted, files with assets to only be included, and BimlScript files. Type in the search textbox to filter Biml files based on their file names. Right click on groups to add Biml files. Right click on items to view, rename, and remove them.



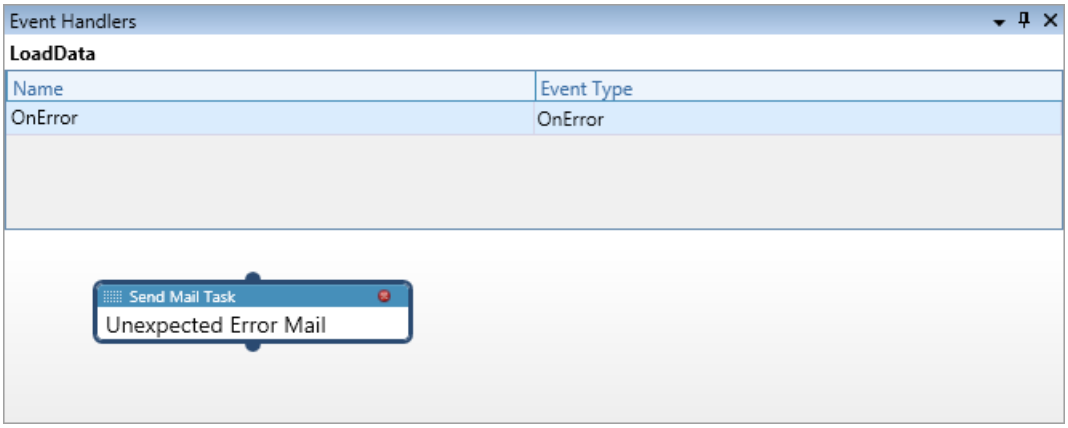
- Properties - Displays all properties for a selected item. Selecting a property in the grid displays a description of the property's purpose.



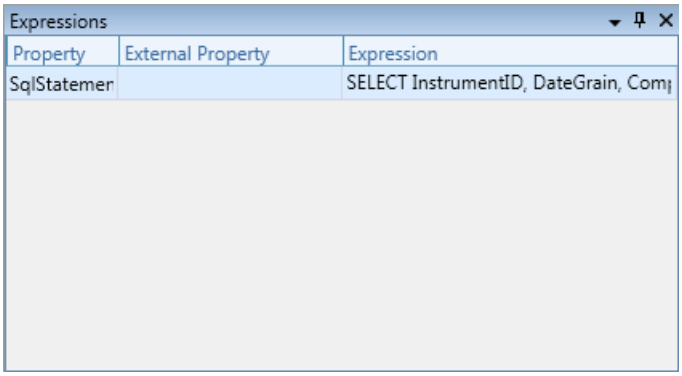
Package Tool Windows Overview

Certain tool windows only appear when a package is opened in the package editor. These tool windows are specifically for editing the properties of packages and their child items. They are listed below in alphabetical order.

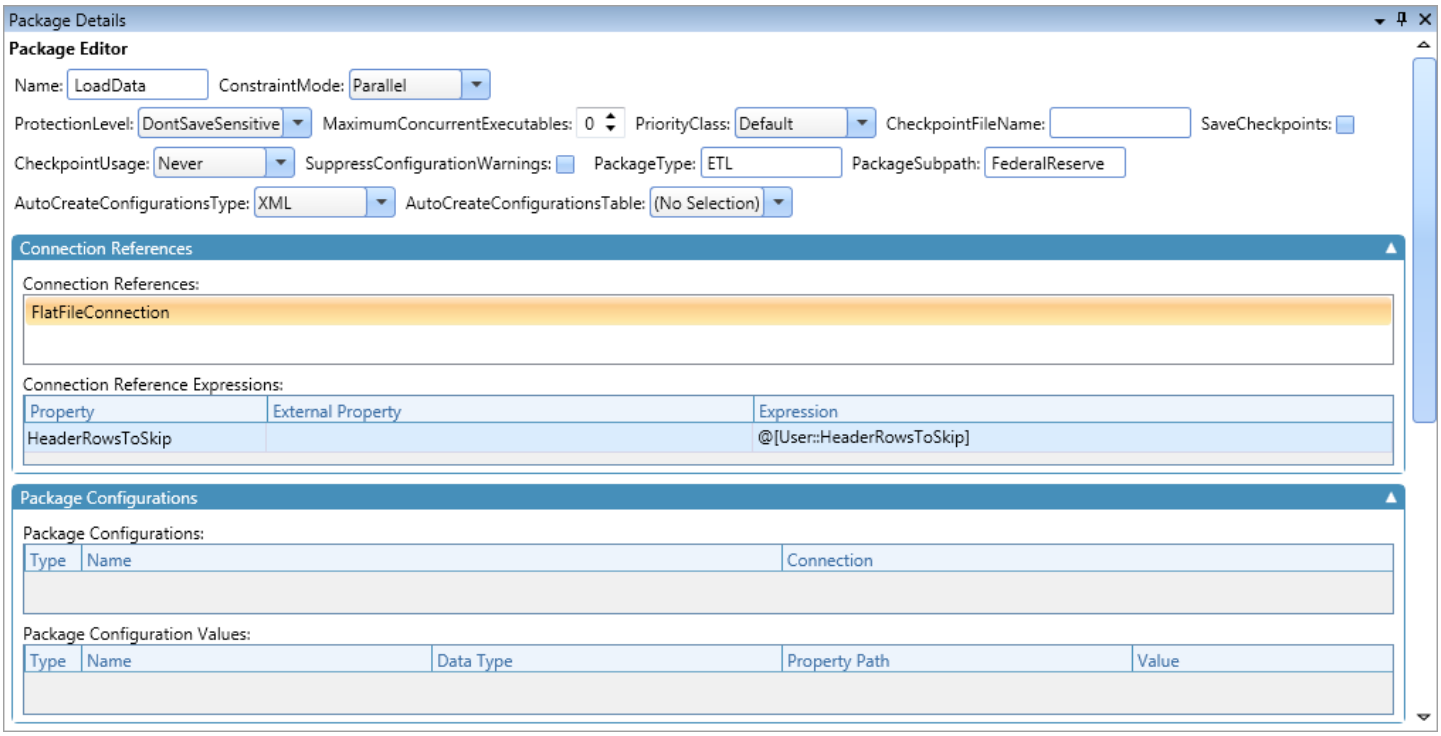
- Event Handlers - Edit the event's name, and modify its type, in the data grid row. Beneath the data grid, use the designer surface to create a workflow for the event.



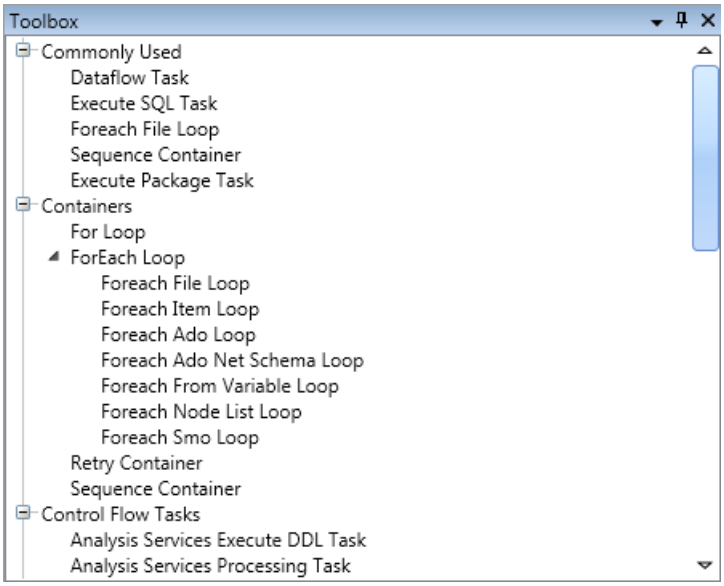
- Expressions - Select an expression's property, or enter an unlisted property, in the data grid row. Use the Expression cell to enter the expression. Use the cell's dialog button to open the Expression Builder dialog for more powerful editing.



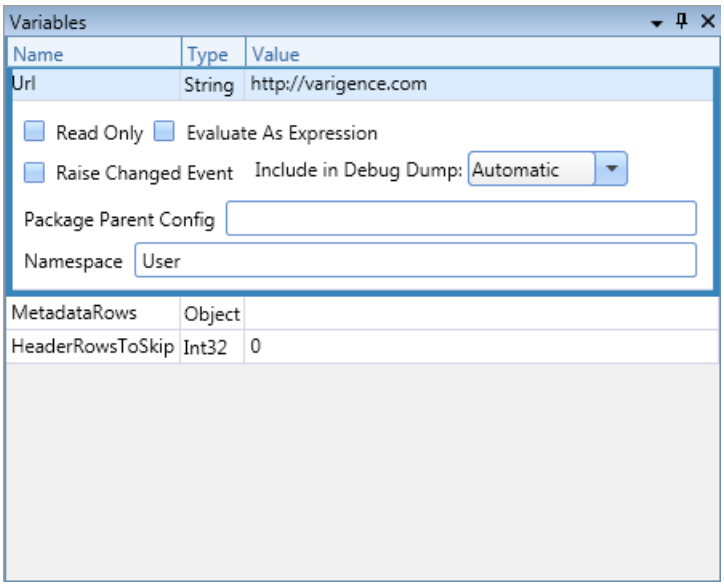
- Package Details - Its content is updated to reflect the unique properties of a selected package, task, transformation, or precedence constraint. View and edit the selected object's properties within this window.



- Toolbox - Lists items that can be added to a selected package, task, or transformation. The list is updated to display the appropriate items for the selected object. Drag and drop an item from the Toolbox to an editor to add it.



- Variables - View and edit the name, type, value, and other properties of a variable.


Variables
Name
Type
Value
Url
String
http://varigence.com
Read Only
Evaluate As Expression
Raise Changed Event
Include in Debug Dump:
Automatic
Package Parent Config
Namespace
User
MetadataRows
Object
HeaderRowsToSkip
Int32
0

Impoprting tables using BimlScript

BimlStudio allows you to import schema and table definitions from a database, along with updating your model with the current state of the definitions at each compile. To leverage the update capability, you will need to use BimlScript to author the import logic; the BimlScript is executed during each compile, obtaining the current state of schemas and tables.

You can import database assets by following these steps:

1. Select the Home tab in the ribbon and click the BimlScript button. You can also right click the Utilities folder and click **Add BimlScript**. This will open the **New Item** dialog, click **Add**.

2. This creates a BimlFile that's added in the Logical View under Library\Utilities.

3. The BimlFle should be open on the right, if not Double click on the BimlFile to open the BimlScript designer.

4. Enter your script in the **BimlScript Input Editor** pane. When executed, the script generates Biml for your imported assets. Clicking on the notification bar saves the script and displays the Biml, resulting from running the script, in the **Preview Expanded BimlScript** pane.

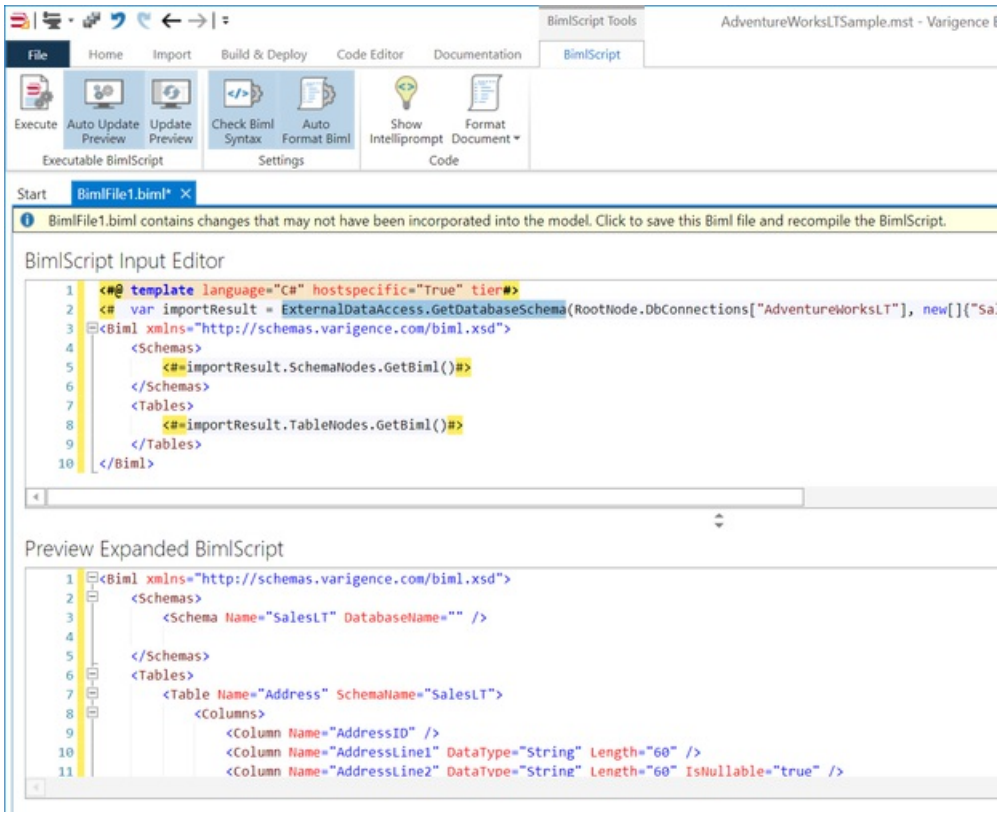
5. For importing assets from the AdventureWorksLT database, you can copy and paste this BimlScript into the **BimlScript Input Editor** pane. You can also use this sample as a starting point for writing your own import script.

```
        <#@ template language="C#" hostspecific="True" tier#>
        <# var importResult =
ExternalDataAccess.GetDatabaseSchema(RootNode.DbConnections["AdventureWorksLT"], new[]{"SalesLT"}, null,
ImportOptions.ExcludeForeignKey | ImportOptions.ExcludeColumnDefault | ImportOptions.ExcludeViews |
ImportOptions.ExcludeIdentity);#>

        <Biml xmlns="http://schemas.varigence.com/biml.xsd">
            <Schemas>
                <#=importResult.SchemaNodes.GetBiml()#>
            </Schemas>
            <Tables>
                <#=importResult.TableNodes.GetBiml()#>
            </Tables>
                <#=importResult.TableNodes.GetBiml()#>
            </Tables>
        </Biml>
```

```
1.  The first thing to notice is that all code fragments are surrounded by required <# and #> characters.
2.  The first line is the required template directive. The language parameter depends on the language you're
using; this example uses C#.
3.  The importResult variable returns a collection of type ImportResults. The property's getter uses RootNode,
which is the root of the project tree; all assets in your project can be reached from RootNode. The
Connections collection returns the _AdventureWorksLT_ connection and used by
ExternalDataAccess.GetDatabaseSchema call method to perform the import.
5.  Within the Biml element, the Schemas and Tables lists are populated by retrieving the SchemaNodes and
TableNodes collections from the Results property and calling GetBiml, which converts the returned schema and
table items into Biml.
6.  To learn more about these functions, check out the Biml API documentation [here.]
(http://www.varigence.com/documentation/biml/api.html)
```

1. While writing the script, you can click on the notification bar to save the BimlScript and examine its output. The Update Preview ribbon button also refreshes the Output editor.

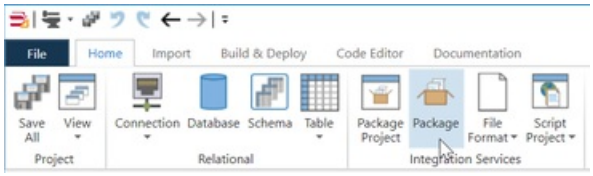


2. Once the output looks correct, press the **Execute** button in the ribbon to import the assets into your model.

Creating a Basic Package

Packages provide data integration functionality. For more information on package properties, see the Package Element.

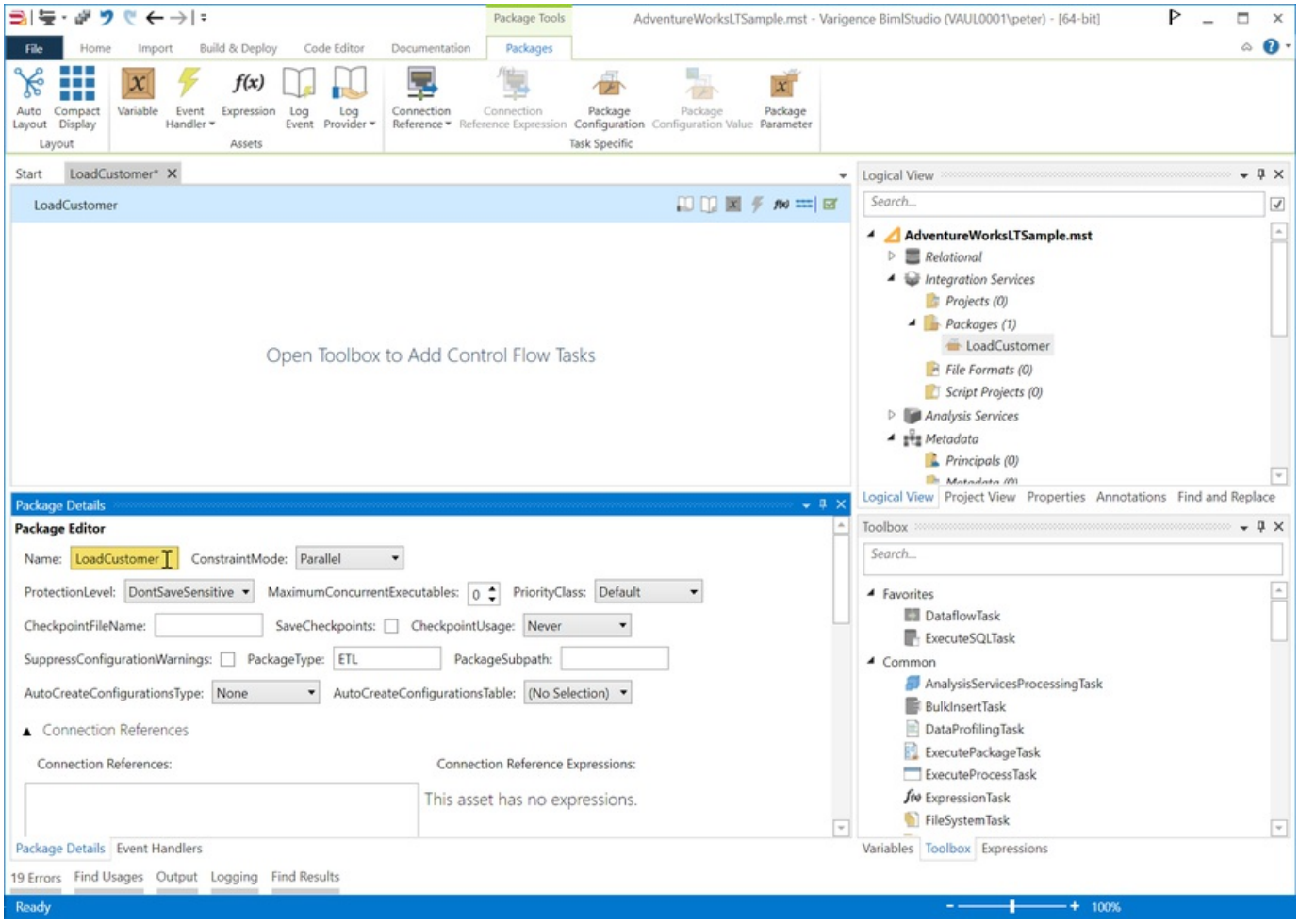
1. New packages can be created from the Home ribbon tab, or from the context menu for packages in the logical view. For this example, select the Package button on the Home tab of the ribbon bar.



2. The package designer should be open on the left, if not Double-click the new package in the logical view to open the package designer.

3. Note the Package Details tool window, which allows you to edit the package properties. This tool window will update with an editor for any selected item on the package, including tasks and components. If the Package Details tool window is not visible, you can show it by selecting the View..Details item from the Home ribbon tab.

4. Change the name of the package using the name field of the Package Details tool window. For the tutorial, name the package "LoadCustomer".



   The AutoCreateConfigurationsType property controls whether configurations are automatically generated for all database connections used in the package. Setting it to None turns generation off, while setting it to XML or SQL generates configurations of the related type at compile time.

5. After adding the package, you should save the project.

Switching Build Types for Biml Files

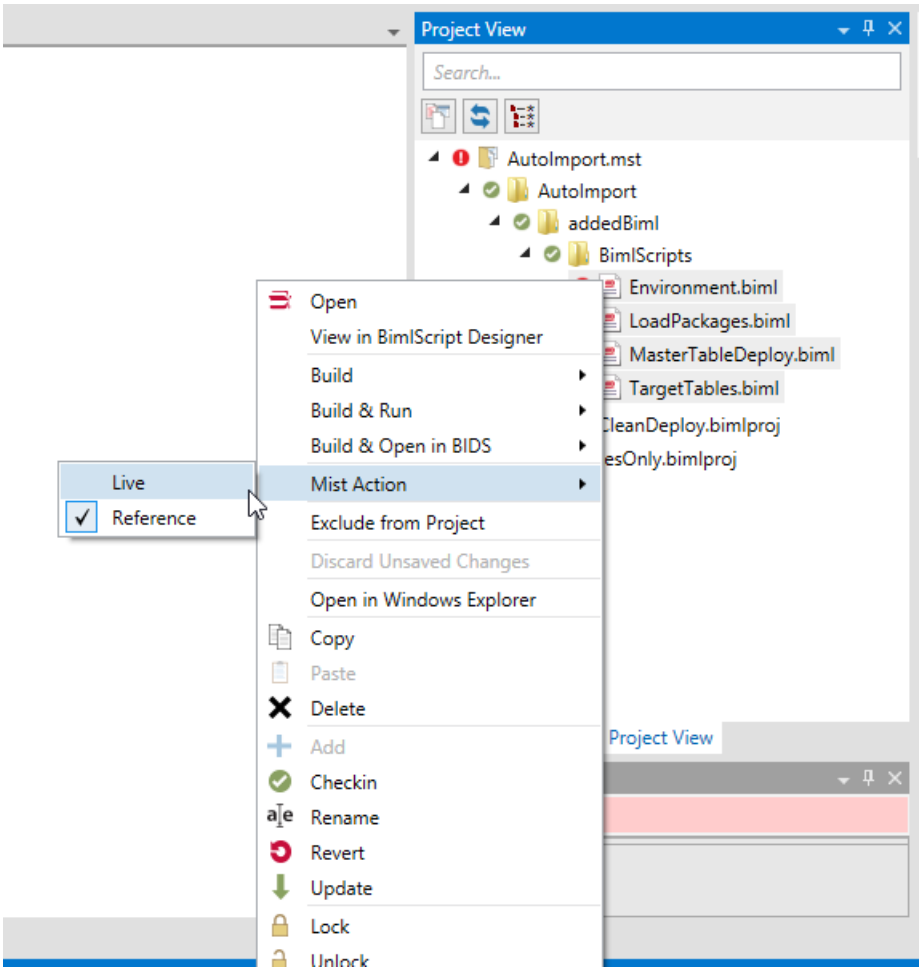
Switching the build type of a Biml file controls whether the file provides only metadata to the compilation process, or if it will produce compiled output. Files marked as Reference are used for metadata or as transformers. Files marked as Live will produce compiled output for the objects they contain.

Note

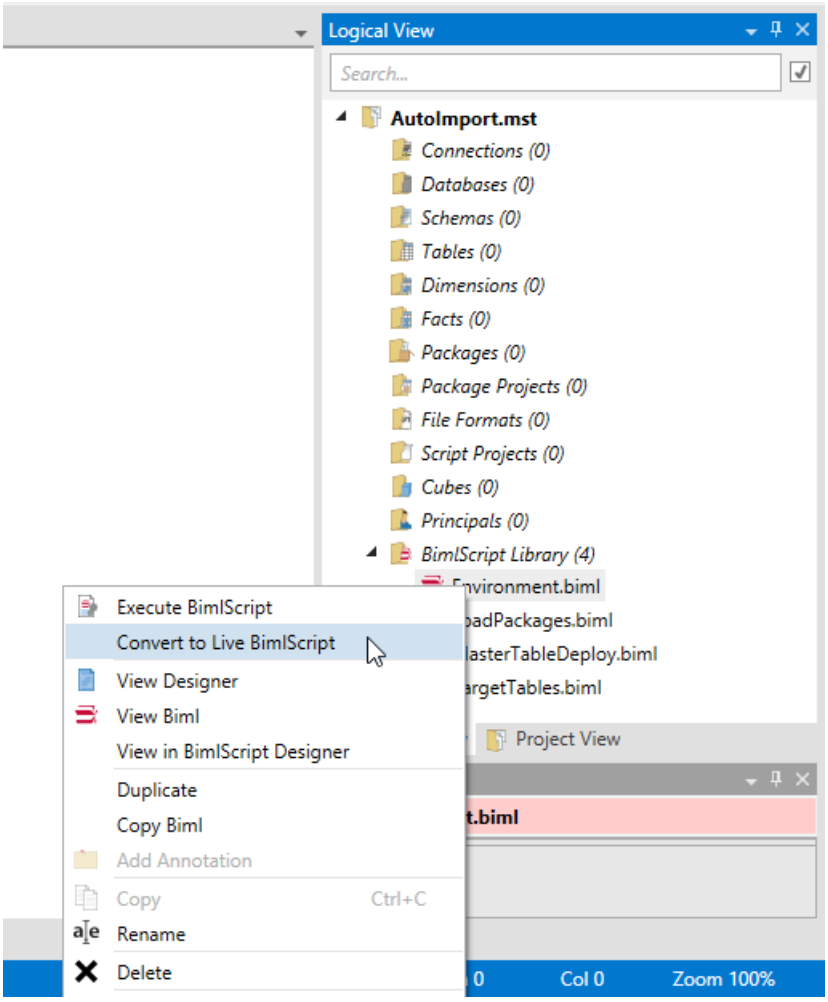
The build type for a Biml file is part of the project settings. A file can be marked Live in one project, and as an Reference in another.

You can switch a Biml file's Biml type by doing one of the following:

- From the Project View, right click on the file, select "Mist Action", and then choose the desired type from the context menu to change the type.



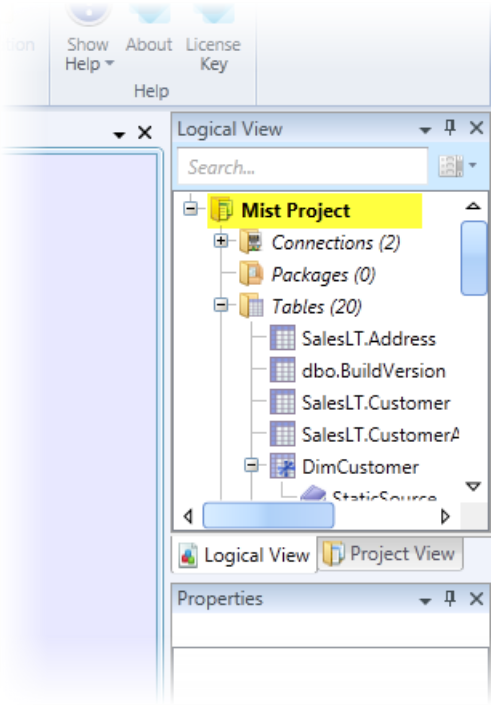
- If a file is already using the reference action, from the Logical View, right click the file under the BimlScript Library, and then click "Convert to Live BimlScript."


Logical View
Search...
AutoImport.mst
Connections (0)
Databases (0)
Schemas (0)
Tables (0)
Dimensions (0)
Facts (0)
Packages (0)
Package Projects (0)
File Formats (0)
Script Projects (0)
Cubes (0)
Principals (0)
BimlScript Library (4)
Project View
Execute BimlScript
Convert to Live BimlScript
View Designer
View Biml
View in BimlScript Designer
Duplicate
Copy Biml
Add Annotation
Copy Ctrl+C
Rename
Delete
Col 0
Zoom 100%

Configuring Project Settings

The project settings editor lets you control project-wide configuration options.

1. Open the project settings editor by double-clicking the project node in the Logical View.



1. See the table below for an explanation of the settings on this page.
2. Save the project after changing settings to commit the updates.

PROJECT SETTING	DESCRIPTION
General	
Hadron Path	The path to the 32-bit version of the Hadron compiler.
64-bit Hadron Path	The path to the 64-bit version of the Hadron compiler.
Extra Command Line Options	Allows additional compiler settings to be passed to the Hadron compiler. See Biml Compiler Command Line Options for details on the available options.
Generate Response File	Creates a response file, based on the current project settings, that can be used for command line builds.
Source Control	
tf Path	The path to the Team Foundation executable. This is only needed if you are using Team Foundation Server for source control.
Versions	
SQL Server	Specifies the version of the SQL Server relational database to target.
SSAS	Specifies the version of SQL Server Analysis Services to target.

PROJECT SETTING	DESCRIPTION
SSIS	Specifies the version of SQL Server Integration Services to target.
Errors and Warning	
Warn As Error	Causes Hadron to treat warnings as errors during the compilation process, so that the build fails if any warnings are encountered.
Output	
Clean Output Folder	Causes the compiler to remove old items from the output folder.
Output Path	Specifies the path where the compiled assets from the project will be stored. This can be a fully qualified path, or a relative path from the project directory.

Hadron Compiler Options

The following Hadron Compiler options are sorted alphabetically.

OPTION	ADDL. FLAGS	PURPOSE	EXAMPLE
--buildOnly	-b	Only the asset with the specified scoped name is emitted.	
--buildOnlyWithDependencies		Only the asset with the specified scoped name, and its dependencies, are emitted.	
--cleanOutputFolder		Delete all files in the output folder before compilation.	
--help	-h -?	Show help.	
--include	-i	Specify a Biml file that needs to be included to build source Biml files.	-i i1.biml -s s1.biml
--option	-o	Additional compiler options that can passed to BimlScripts.	
--packageConfigurationPath		Specify the path for SSIS Xml Package Configuration files.	
--responseFiles	-r -@	Specify a response file for compilation.	
--source	-s	Specify a Biml file to compile and emit.	-i i1.biml -s s1.biml
--targetPath	-t	Specify the output directory for the generated files.	
--transformationScriptResourceAssembly		Specify a custom resource assembly with Transformer BimlScript files.	
--transformationScriptSettings		Specify a custom Transformer BimlScript settings file.	
--version	-v	Set a version for SQL/SSIS/SSAS. Versions may be prefixed with Sql/Ssis/Ssas.	--version=Ssas2008
--warnAsError		Treat warnings as errors during build, causing compilation to fail.	
--workflowPath	-w	Specify the directory containing Hadron workflow files.	

Source Control Setup

Mist has the ability to work with projects that are stored in TFS and Subversion source control environments. To work with either source control environment, some additional setup is required.

Subversion

1. Install CollabNet's Subversion client, which can be downloaded from CollabNet Subversion downloads.

- Neither SlikSVN nor other Subversion clients are currently supported.

1. Install TortoiseSVN from TortoiseSVN downloads.

- Do **not** install the command line client tools when installing Tortoise SVN.

1. Ensure paths to svn.exe (installed with CollabNet's Subversion client) and tortoiseproc.exe (installed with TortoiseSVN) are in your Path environment variable.

2. Add your Mist project to your Subversion working copy.

    - If no working copy is present, then it must be created. See the Tortoise documentation for more information.

3. Open the project in Mist.

TFS

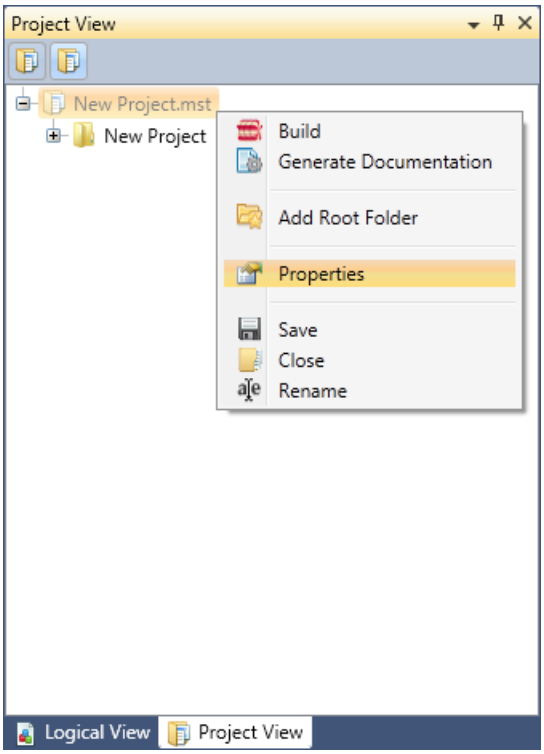
1. Ensure tf.exe is available on the machine where Mist is installed.

- tf.exe is not included in VS 2008 installations. It can be downloaded from theVisual Studio Team System 2008 Team Explorer package.
- tf.exe is included, by default, in VS 2010 installations.

1. Install the Team Foundation Server Power Tools, which can be downloaded from Visual Studio Power Tools
2. Add your Mist project to a TFS workspace.

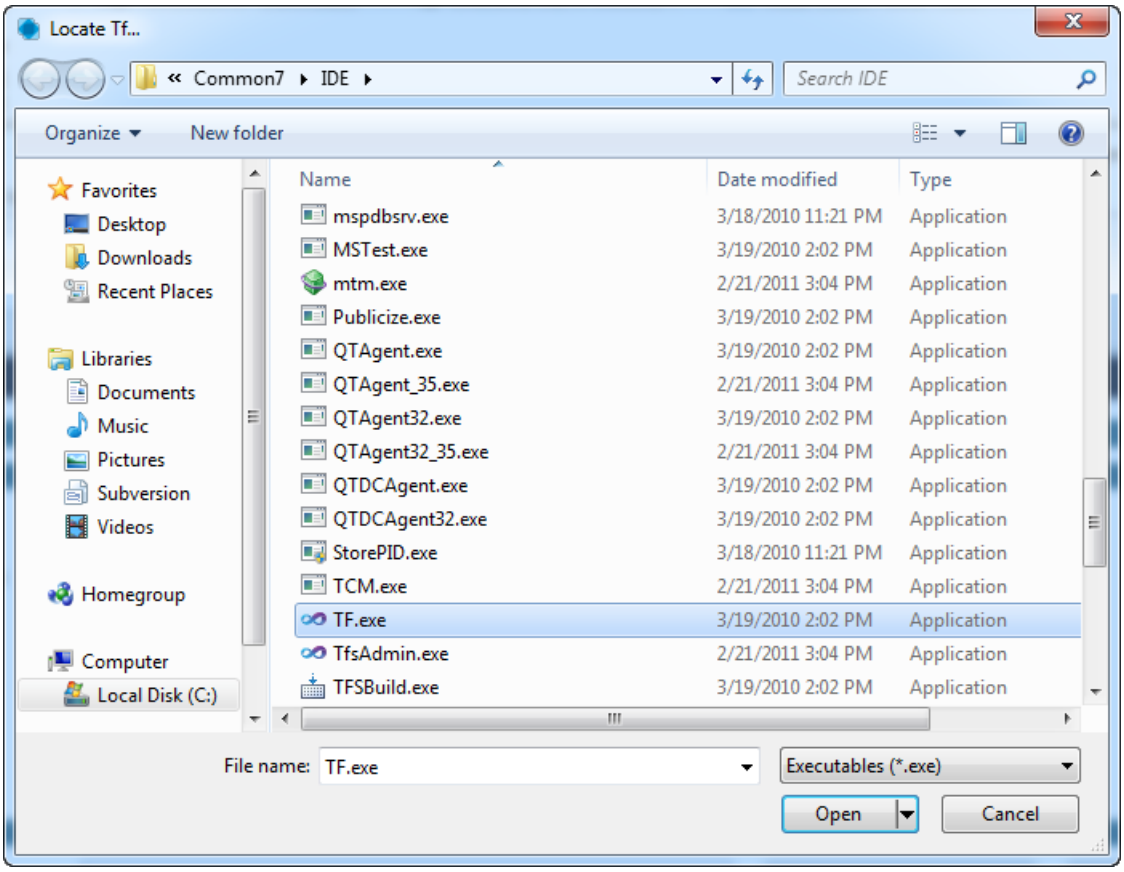
- To create a TFS workspace that contains your project, see the MSDN documentation for Visual Studio 2008 and Visual Studio 2010 users.
- To add files and folders to TFS version control, see the MSDN documentation
- For those just getting started with TFS, MSDN provides detailed documentation on how TFS works and how to use it.

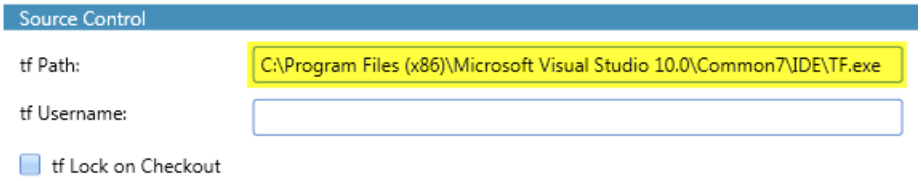
1. Open the project in Mist.
2. In Project View, right click on the project and select **Properties**.



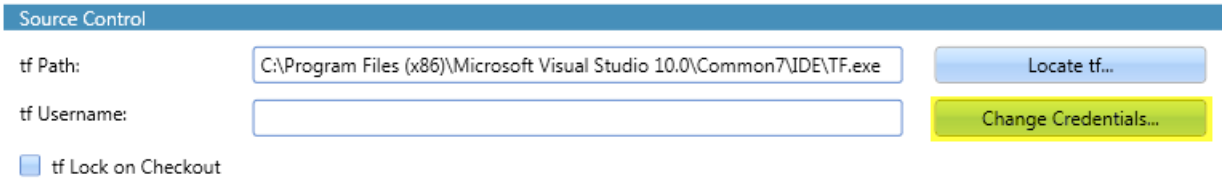
1. In the Properties designer, press the **Locate tf...** button.

1. Find tf.exe on your system and press **Open**. The path to tf.exe will be automatically inserted in the tf Path textbox.

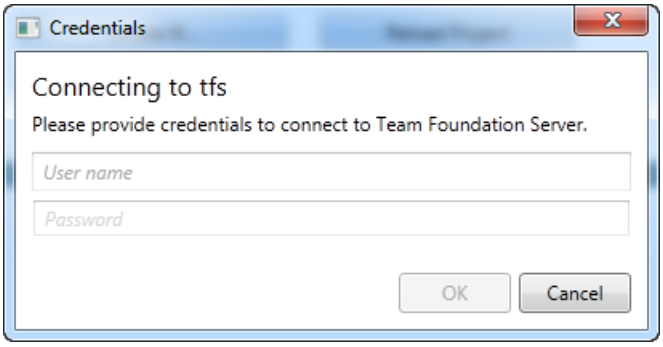




1. If your TFS enlistment requires logon credentials, press the **Change Credentials** button.



1. Enter your username and password in the Credentials dialog box and press **OK**.

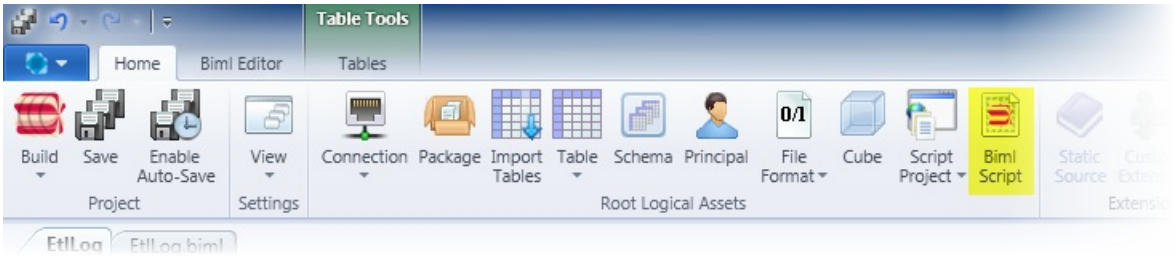


1. Click the **Reload Project** button.

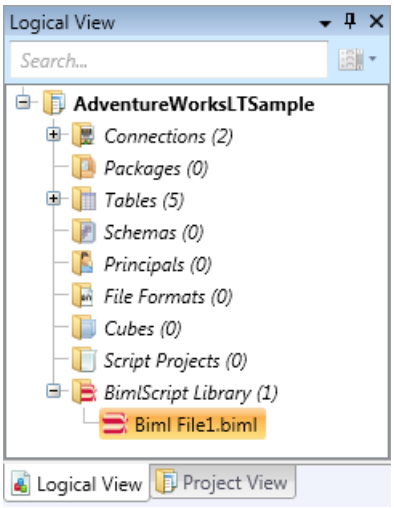
Create an Expandable Transformer

Expandable Transformers are BimlScripts let you apply modifications across a large number of Biml objects.

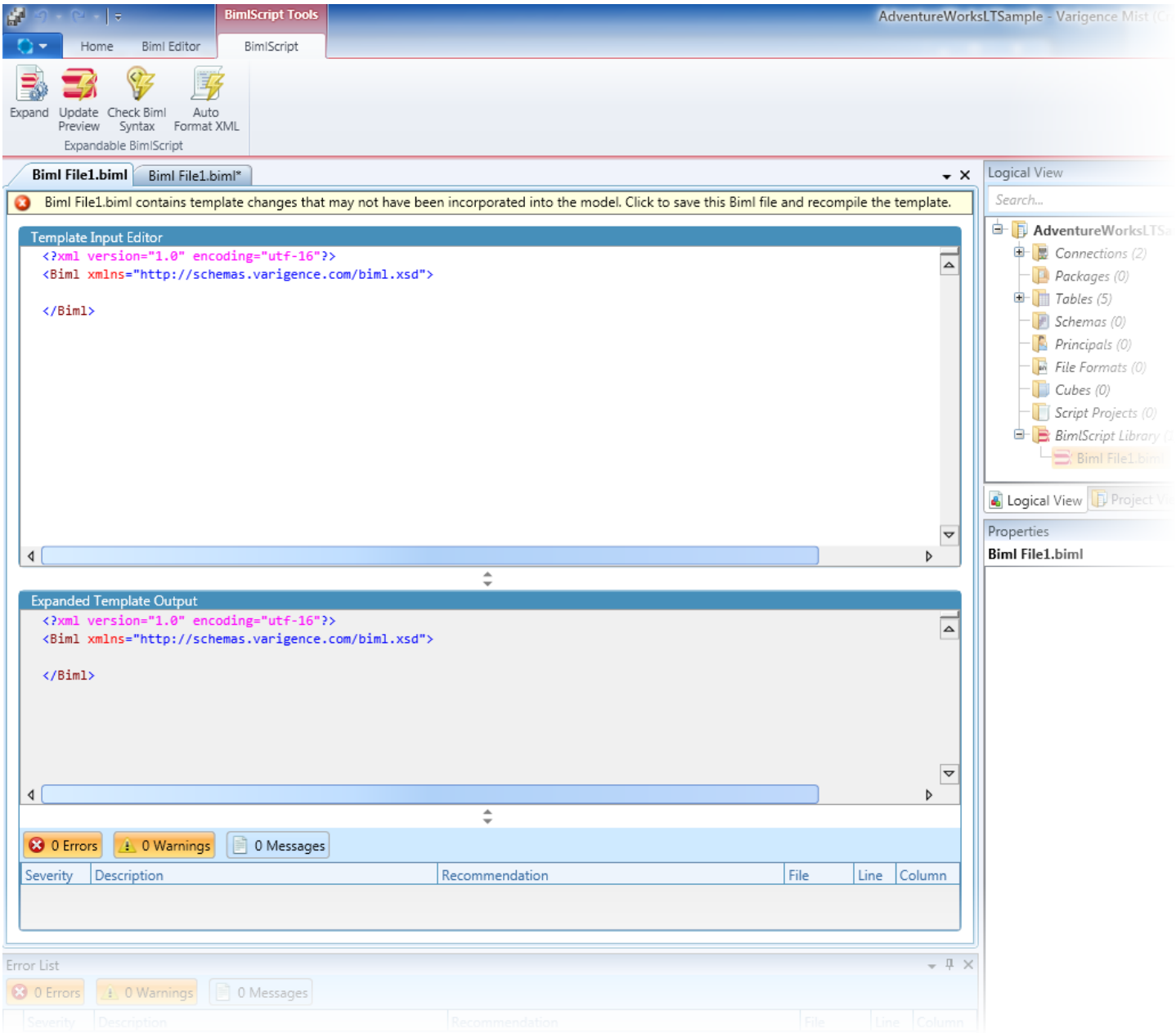
1. To create an Expandable Transformer, you need a BimlScript. BimlScripts can be created from the Home ribbon tab, or from the context menu for packages in the logical view. For this example, select the BimlScript button on the Home tab of the ribbon bar.



2. This creates a Biml file that's added in the Logical View under BimlScript Library.



3. Double click on the Biml file to open the BimlScript designer.



4. Enter your BimlScript in the Input editor. Remember that Expandable Transformers are intended to edit all Biml elements of a specific type.

5. One possible use of Expandable Transformers is to add a variable to all packages in your project. To try this, you can copy and paste this BimlScript into the Input editor. Also, add a couple packages to your project so you can see the effect of the transformer. Note that you can also use this sample as a starting point for writing your Expandable Transformers.

```
<#@ template language="C#" hostspecific="True" addtoheadofcollections="True" #>
<#@ target mergemode="LocalMerge" type="Varigence.Languages.Biml.Task.AstPackageNode" exactmatch="true" #>
<#@ import namespace="Varigence.Languages.Biml.Task" #>
<Package>
<Variables>
<Variable Name="ID" DataType="UInt32"/>
</Variables>
</Package>
```

1. The first thing to notice is that all code fragments are surrounded by required <# and #> characters

2. The first line is the required template directive. The language attribute dictates the language you're using; this example uses C#.

3. Import directives specify namespaces, and are analogous to the C# using statement or the VB.NET imports statement.

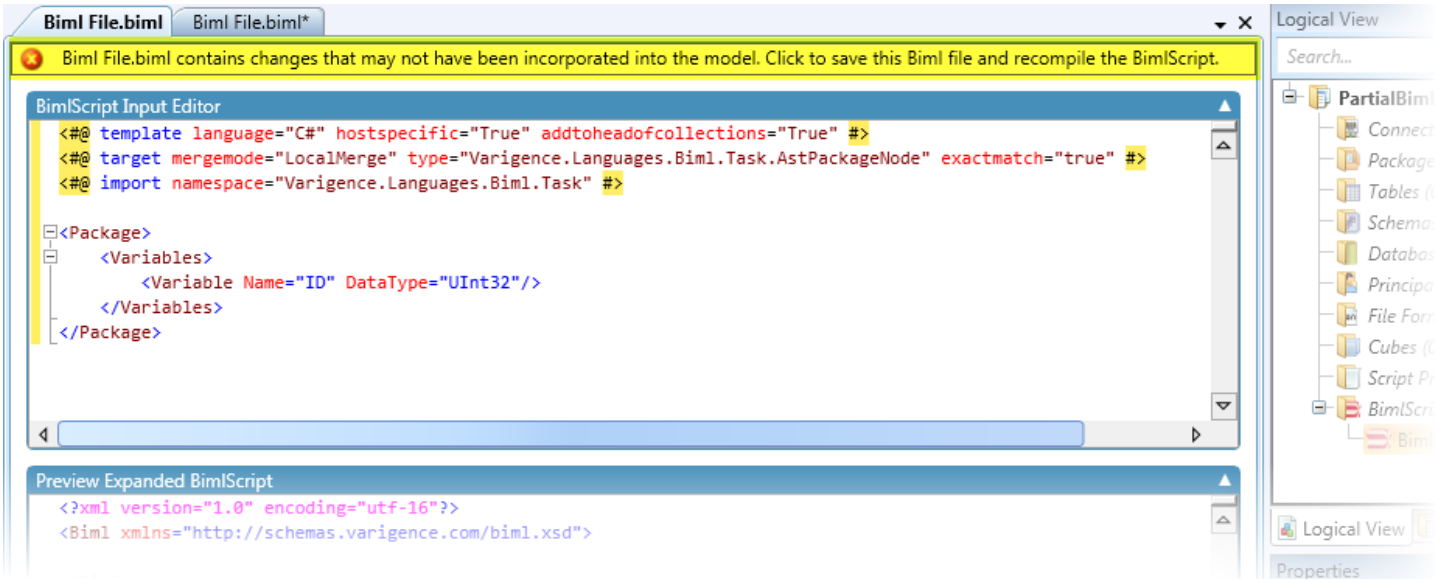
4. For transformers, you need to specify the type of Biml asset to change. That's accomplished using the target directive's type attribute. The example's target type is AstPackageNode, indicating the transformer can run on all Package nodes in the project.

5. The target directive's exactmatch attribute controls if the transformer should run only on Package nodes, as opposed to including any types derived from AstPackageNode. Because exactmatch is set to true, this transformer only runs on Package nodes.

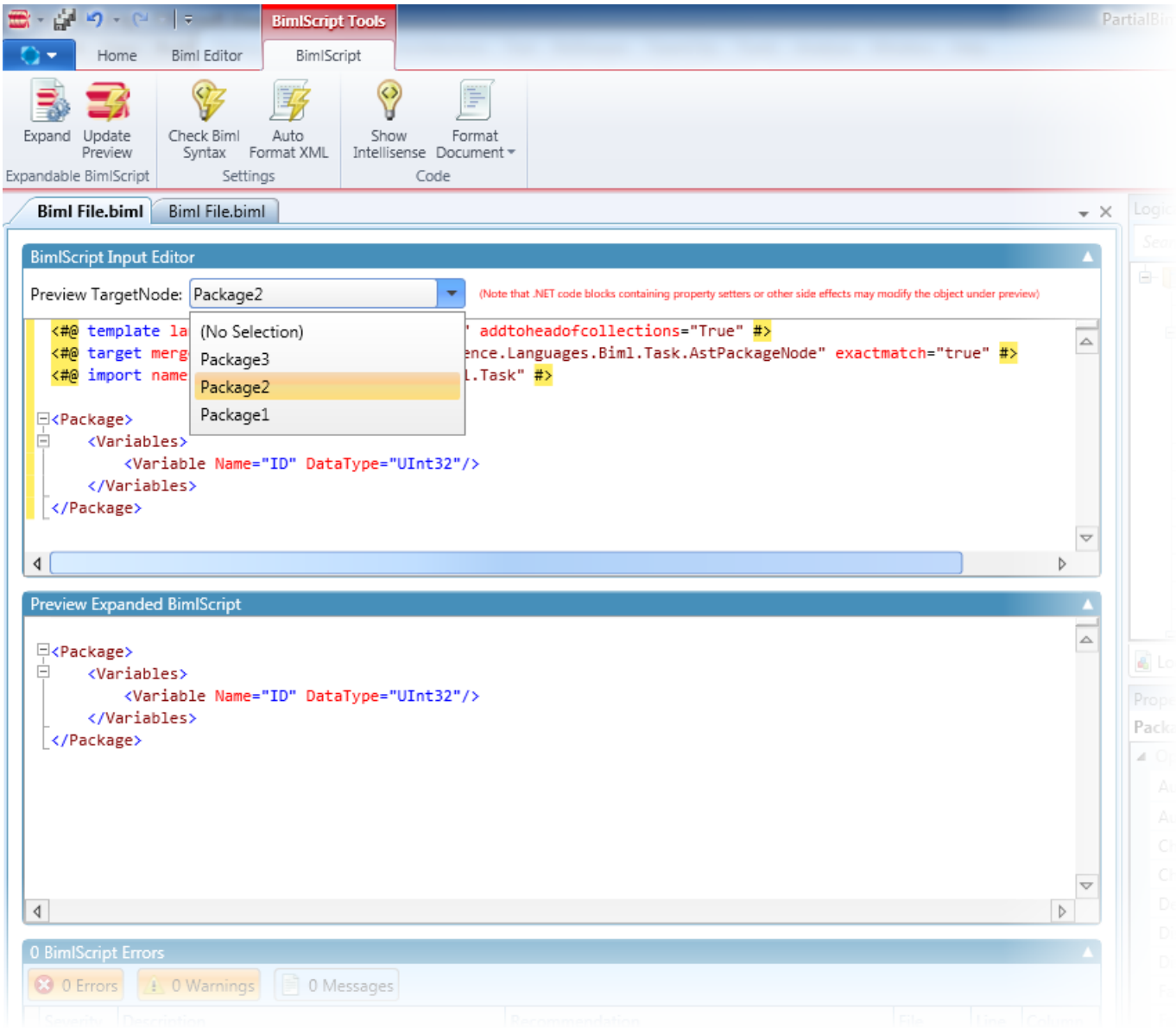
6. The target directive's mergemode attribute controls how the specified changes are merged into the existing Biml. By specifying a mergemode of LocalMerge, the transformer will merge the specified variables with any variables that already exist in a Package.

7. The template directive's addtoheadofcollections attribute controls how the assets, declared within collections, are merged into a node's already existing collections. By setting this to true, the variable declared in the transformer will be inserted at the head of any Package's variables collection.

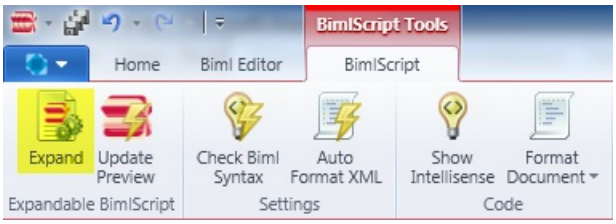
6. While writing the script, you can click on the notification bar to save the BimlScript.



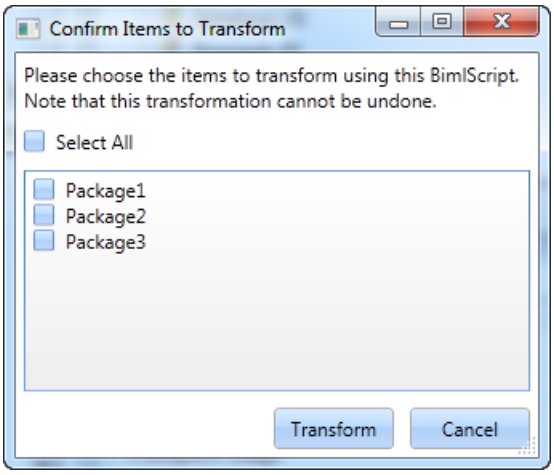
7. To preview the transformer's output on a particular Package, select a Package from the Preview Target Node dropdown. Assuming the BimlScript has no errors, Preview Expanded BimlScript pane will display the new Biml that would result from running the transformer.



8. Once the output looks correct and you're ready to apply the changes, press the Expand button in the ribbon.



9. The last step before applying the transformer is to select which assets to transform. Using the above sample, all Packages in the project are listed. You can check any package, or check Select All to apply the transform to all of them. When finished, click the Transform button to apply the transformer.



10. All transformed packages now include the ID variable that was defined in the Expandable Transformer.

Choose the Right BimlScript Type

BimlScript is useful in a variety of situations, including those that require automation (such as repetitive tasks) or where the declarative Biml model isn't sufficient; e.g. conditionally creating a column in a table.

Before you write a BimlScript, however, you'll first want to decide which BimlScript type is appropriate for your task.

There are three BimlScript types to choose from:

1. Executable
2. Live
3. Transformers

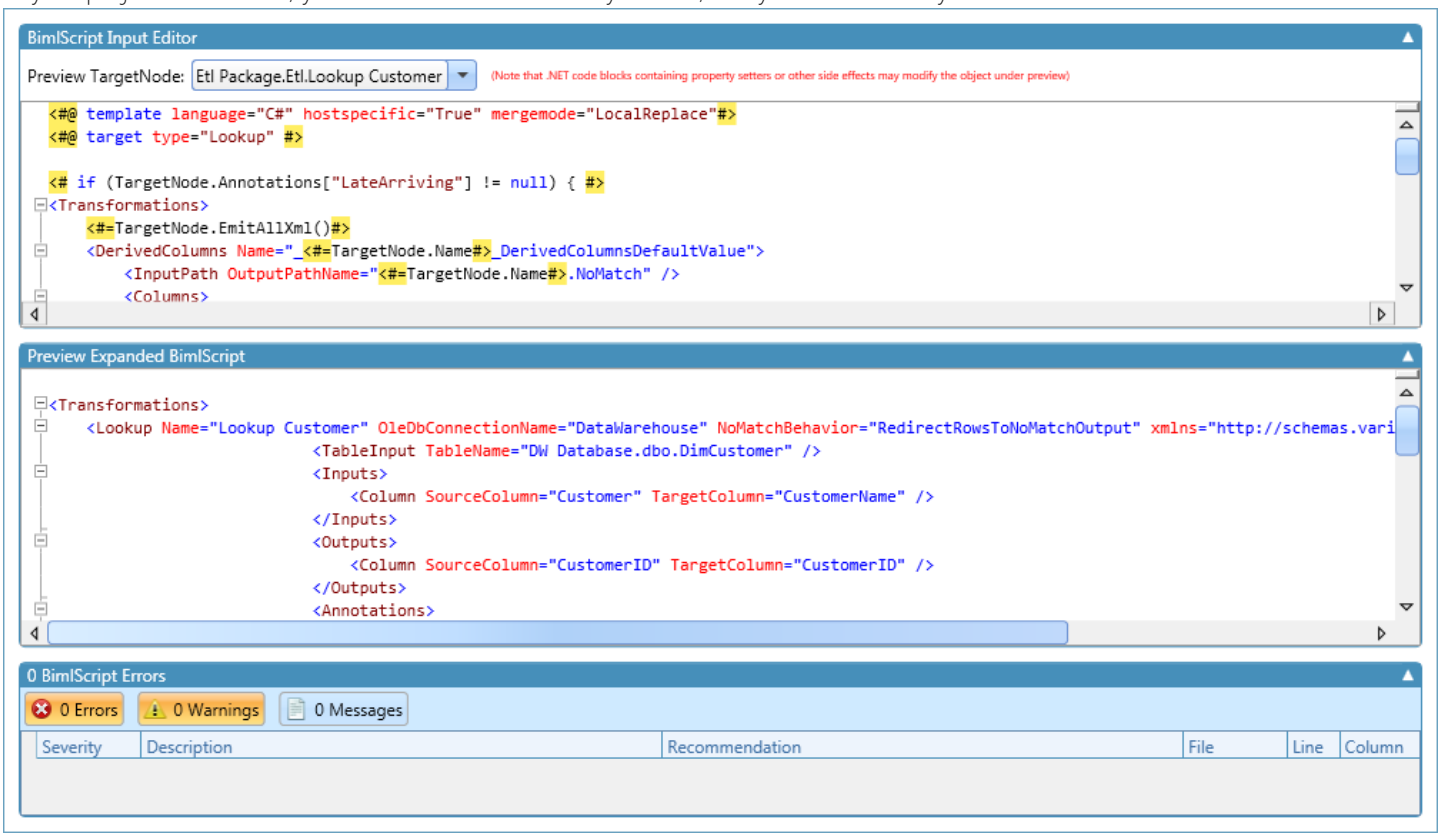
This article explains each type, enabling you to choose the right type for your needs.

Executable

Trigger: On Demand:

Description: Executable BimlScript is BimlScript that is applied to a Mist project on-demand. Executable BimlScripts are created by adding BimlScript assets to a Mist project; these assets are listed in the logical view's BimlScript Library group. An executable BimlScript can be authored in Mist's BimlScript editor; the BimlScript's results can be previewed within the editor too. When the BimlScript is error free, you can press the ribbon's Expand button to compile and execute the BimlScript. This results in any assets, created by the executable BimlScript, being immediately added to your Mist project.

Example Creating New Objects: Our Importing Tables With BimlScript example demonstrates executable BimlScript. Each time you press the Expand button, the BimlScript is compiled and any .NET code is executed. All assets generated by the code are then added to your project. From there, you can alter them however you wish, as if you had manually created each one.



To demonstrate this, consider a project that has an ETL package with a couple of Lookup components. We want to update each Lookup to handle Late Arriving Dimension logic. To accomplish this, we need a way to define the Lookups in Packages that already exist.

An executable BimlScript, that inserts Late Arriving Dimension logic, might look like this:

```
<#@ template language="C#" hostspecific="True" mergemode="LocalReplace"#>
<#@ target type="Lookup" #>

<# if (TargetNode.Annotations["LateArriving"] != null) { #>
<Transformations>
    <#=TargetNode.EmitAllXml()#>
    <DerivedColumns Name="_<#=TargetNode.Name#>_DerivedColumnsDefaultValue">
        <InputPath OutputPathName="<#=TargetNode.Name#>.NoMatch" />
        <Columns>
            <Column Name="<#=TargetNode.Annotations["LateArriving"].Text#>" DataType="Int32">0</Column>
        </Columns>
    </DerivedColumns>
    <OleDbCommand Name="_<#=TargetNode.Name#>_OleDBCommandInsertPlaceholderRow"
ConnectionName="DataWarehouse"
        ValidateExternalMetadata="False">
        <DirectInput>EXEC pInsertCustomer ?, ? OUTPUT</DirectInput>
        <Parameters>
            <Parameter SourceColumn="Customer" TargetColumn="@CustomerName" />
            <Parameter SourceColumn="SalesAmount" TargetColumn="@CustomerID" />
        </Parameters>
    </OleDbCommand>
    <UnionAll Name="_<#=TargetNode.Name#>_UnionAllMatchAndNoMatch">
        <InputPaths>
            <InputPath OutputPathName="<#=TargetNode.Name#>.Match" />
            <InputPath OutputPathName="_<#=TargetNode.Name#>_OleDBCommandInsertPlaceholderRow.Output" />
        </InputPaths>
    </UnionAll>
</Transformations>
<# } #>
```

The first thing to notice is that the Biml portion doesn't start with a Biml tag; it begins with a Transformations tag instead. This BimlScript fragment, that doesn't begin with a Biml tag, is called a *partial* BimlScript. Partial BimlScripts are used to merge changes into existing Biml assets. Only Executable and Transformer BimlScripts support partial BimlScript.

This partial BimlScript generates a Biml fragment that will be inserted into existing nodes. The type of node that will be edited is of type Lookup, which is set in the target type directive at the top of the script. When this partial BimlScript is executed via the ribbon's Expand button, it will be applied to all Lookup nodes in the project, inserting the Transformations collection that's specified in the script.

By specifying a target type, BimlScripts gain access to a built-in TargetNode property. When a BimlScript is executed, it's run on each node in the project whose type matches the target type. The TargetNode property contains the node that the BimlScript is being run on. This enables a BimlScript author to set conditionals based on the properties on a node; in the above example, there's an if statement that checks if the TargetNode's Annotations collection holds an annotation named 'LateArriving'. This capability enables fine grained control of the specific nodes that will be modified by a BimlScript.

Live

Trigger: Project Changes

Description: Live BimlScript is BimlScript that's placed within a regular Biml file. This causes the BimlScript-generated objects to become 'live' objects. They are compiled and executed whenever there are changes to the project. Thus, assets generated by live BimlScript are always up-to-date. Because they are re-generated by any change, however, they can't be altered within Mist visual designers; changes must be made to the live BimlScript itself, using the Biml editor.

Example Creating New Objects One common use of live BimlScript is automatically generating Biml code within a Biml file. For instance, the Federal Reserver sample demonstrates live BimlScript in the LoadDimDate package. Within this package is an ExecuteSQL task that needs to insert all dates, from January 1st 1900 to December 31st 1979, into a Date dimension. Rather than generating the dates in another tool and copy/pasting them into the editor, live BimlScript is used. A simple for loop iterates over the range of dates,

adding an INSERT INTO statement for each. This is particularly powerful if you ever need to change the date range; a one line change to enter the new date causes the live BimlScript to immediately re-run, generating the updated dates in the SQL query.

Of course, live BimlScript can also be used to generate Biml assets.

```
<#@ template language="C#" hostspecific='true'#>
<#@ import namespace="System.Data" #>

<Biml xmlns="http://schemas.varigence.com/biml.xsd">
    <Connections>
        <OleDbConnection Name='Source' ConnectionString='Provider=SQLNCLI10;Server=.;
            Initial Catalog=AdventureWorksDW2008R2;Integrated Security=SSPI;'/>
    </Connections>
    <Packages>
        <Package Name='Rebuild Warehouse Data' ConstraintMode='Linear' AutoCreateConfigurationsType="None">
            <Tasks>
                <# foreach (var table in RootNode.Tables)
                {
                    if (!table.Schema.Name == 'dbo')
                    {
                    #>
                        <Dataflow Name="Copy Data" <#=table.Name#>>
                            <Transformations>
                                <OleDbSource Name="Retrieve Data" ConnectionName="Source">
                                    <DirectInput>SELECT * FROM <#=table.Name#></DirectInput>
                                </OleDbSource>
                                <OleDbDestination Name="Insert Data" ConnectionName="Target">
                                <ExternalTableOutput Table="<#=table.Name#>"/>
                                <OleDbDestination>
                            </Transformations>
                    </Dataflow>
                    <# }
                } #>
            </Tasks>
        </Package>
    </Packages>
</Biml>
```

In this example, a new Dataflow task is created for each table in a project that is not in the dbo schema. Each Dataflow task retrieves a table's data from a Source connection and copies that data to a Target connection. This illustrates that BimlScript can take advantage of conditionals like any .NET code.

If the above sample had been applied as executable BimlScript, and new tables were added in the project, Dataflow tasks for those tables wouldn't be automatically added to the package. This illustrates the key difference between executable vs. live BimlScript: one-time generation of editable assets vs. always up-to-date and un-editable assets.

Transformers

Trigger:

Build

Description The final type of BimlScript is a transformer. Transformers are BimlScripts whose changes are only applied when building a project, instead of the changes being applied within Mist.

Example: One scenario where transformers are useful is when an architect wants to apply a specific pattern to Biml assets used within a project, or even across multiple projects. For instance, perhaps there's a logging pattern that should be applied to all packages. The architect can author the pattern in a transformer within Mist, letting him leverage intelliprompt and the BimlScript editor's preview pane. However, instead of executing it in Mist (which would only apply it to the current set of packages), he can add a command line

option to the project to execute the transformer's Biml file during a build:

--transformationScriptSettings <path to ScriptSettings file> *the ScriptSettings file contains a path to the BimlScript file*

When another developer builds the project, the logging pattern is applied to all of his project's packages automatically. Even better, if the pattern needs to be updated in the future, the architect can make the appropriate change and any developer can apply it just by rebuilding his project.

Transformers allow architects to worry about enforcing patterns and best practices, while other developers focus on the business logic. Additionally, transformers make it easy to change patterns over time by not altering the Biml files on disk.

Conclusion

As discussed, the three types of BimlScript enable a wide range of development scenarios. Additionally, partial BimlScripts empower you to merge changes into existing Biml assets.

This chart summarizes the three types of BimlScript and their key differences. Refer to it as you begin leveraging BimlScript so you can choose the right type for your scenario.

TYPE	MAIN PURPOSE	TRIGGERED BY	WHERE DEFINED
Executable	Producing one-time assets that can then be edited further.	Expand button	BimlScript Library
Live	Producing always up-to-date assets (that can't be manually changed).	Any project change; always running.	Within any Biml file.
Transformer	Enforcing development patterns during builds.	Build	BimlScript Library & Plugins

Using Configuration Files

Introduction

Mist 3.0 introduces configuration files as a new, powerful means of creating customized builds. This article explains what configurations files are, and how to:

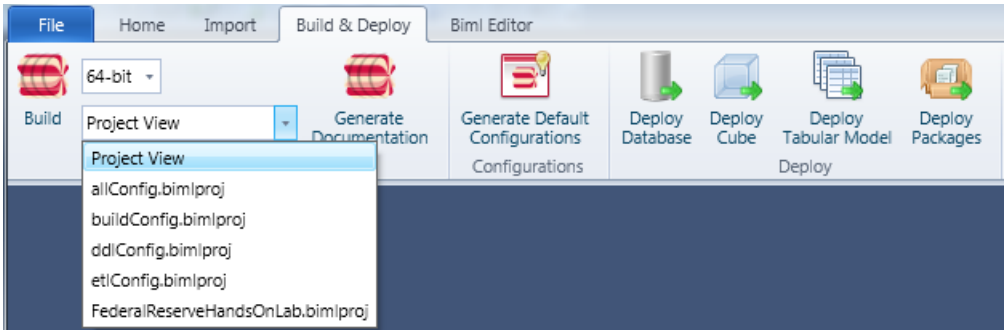
- Generate default configurations
- Create custom configurations
- Use the Project View configuration

Building in Mist

Prior to Mist 3.0, building a project required a user to assign an Emit Type to each Biml file in their project. Biml files that should produce compiled output for their assets were marked as emittables. Biml files that should provide metadata to the compilation process were marked as includes. A user would then store the mapping of Biml files to their emit types as a configuration. When building, the user would select which configuration they wanted to build.

Starting with Mist 3.0, a user can leverage configurations to do far more than assign emit types to Biml files. Configurations allow users to control the order Biml files are built, refactor their build processes, execute commands before and after a build, and more.

To build a Mist project, select the ribbon's Build & Deploy tab. Within this tab, a user can choose a 32 or 64 bit build, select the desired build configuration, and begin a build by pressing the Build button.



When initiating a build, the selected configuration file is processed. Within each configuration file is a build configuration, describing which Biml files to compile and which Biml files should provide metadata. Thus, the emit type information that used to be stored solely in Project View has now been moved into configuration files. Adding this layer of indirection is what enables far more customized and powerful builds in Mist.

Configuration Files

At its core, a configuration file is a MSBuild file with a bimlproj file extension. MSBuild is the Microsoft Build Engine, which includes an executable that uses MSBuild files to build applications, DLLs, and other .NET assemblies. MSBuild files are human-readable XML files that provide the build engine with the necessary information to build the desired assembly.

Mist has extended MSBuild to use the Hadron compiler to build BI assets. The result is that users can leverage MSBuild's reliable and well supported build engine to create fully customizable builds.

As aforementioned, configuration files list the Biml files to be built, along with associating each file with a build type. However, by leveraging MSBuild, it's also possible for configuration files to reference other configuration files during a build, or to execute arbitrary commands before and after a build occurs.

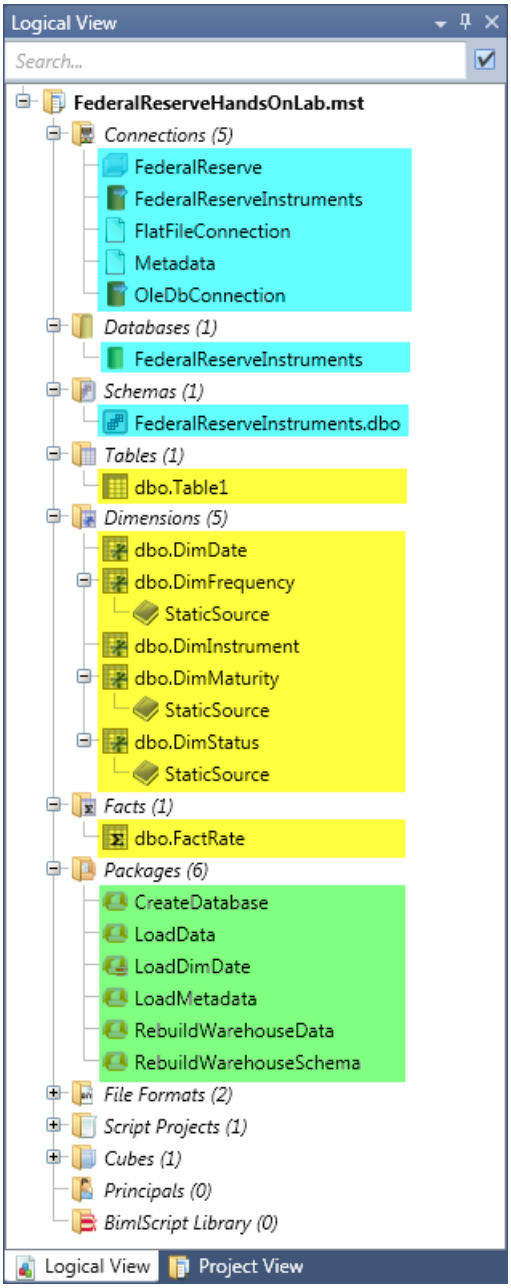
To make dealing with configurations easy, Mist provides a visual designer for setting typical configuration properties. Thus, there is no need to directly edit MSBuild files in a majority of cases.

Generating Default Configurations

Mist can auto generate configuration files that organize your project's Biml files in pre-determined ways. When generating default configurations, Mist adds four configuration files to your project:

FILE NAME	DESCRIPTION
ddlconfig.bimlproj	Generates assets from all Biml files that contain Table, Schema, Database, and Connection assets.
etlconfig.bimlproj	Generates assets from all Biml files that contain Package and Connection assets. Biml files containing Schema and Database assets provide metadata.
build.bimlproj	Builds the Biml files in ddlconfig.bimlproj, followed by the Biml files in etlconfig.bimlproj.
all.bimlproj	Generates assets from all Biml files under the project's directory.

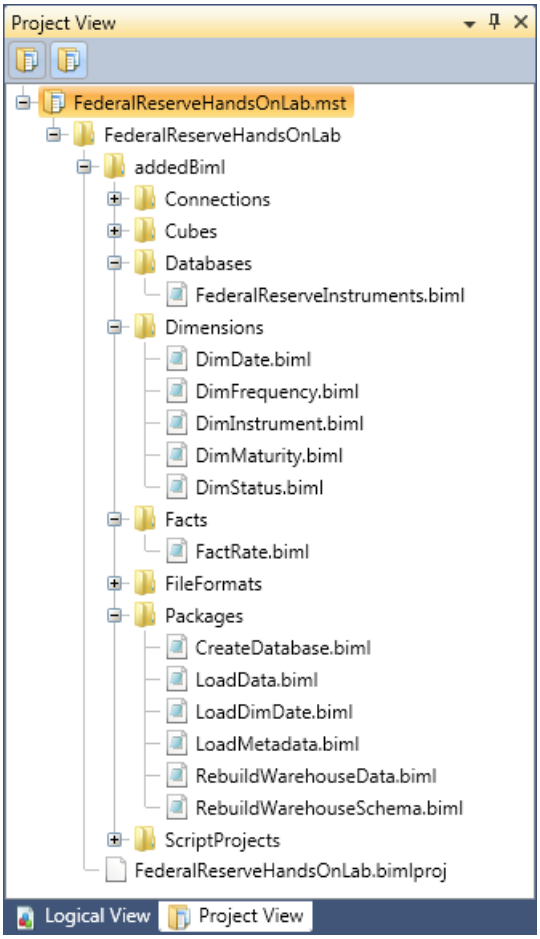
To better understand the contents of configurations, let's review an example from a sample Mist project. This project's logical view comprises of:



The Biml files, that contain the yellow highlighted assets in the logical view, are only added to DDL configurations. The Biml files, that contain the green highlighted assets, are only added to ETL configurations. The Biml files, containing the blue highlighted assets, are added to DDL and ETL configurations.

Creating default configurations

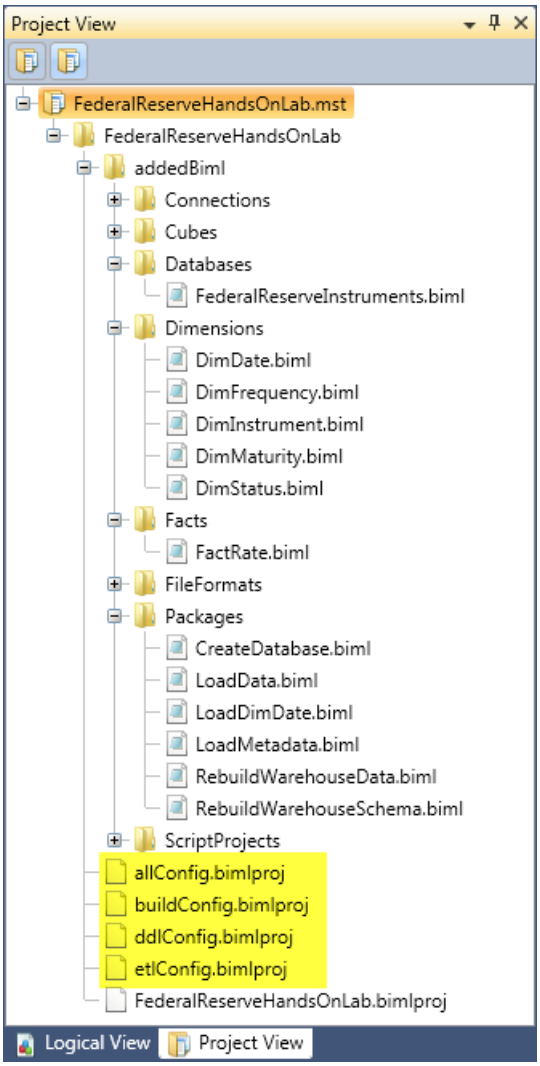
Our sample project contains the following files:



Notice that one configuration file is already included; Mist automatically adds a configuration file to all projects.

To generate the default configurations, select the Build & Deploy ribbon tab and press the Generate Default Configurations button.

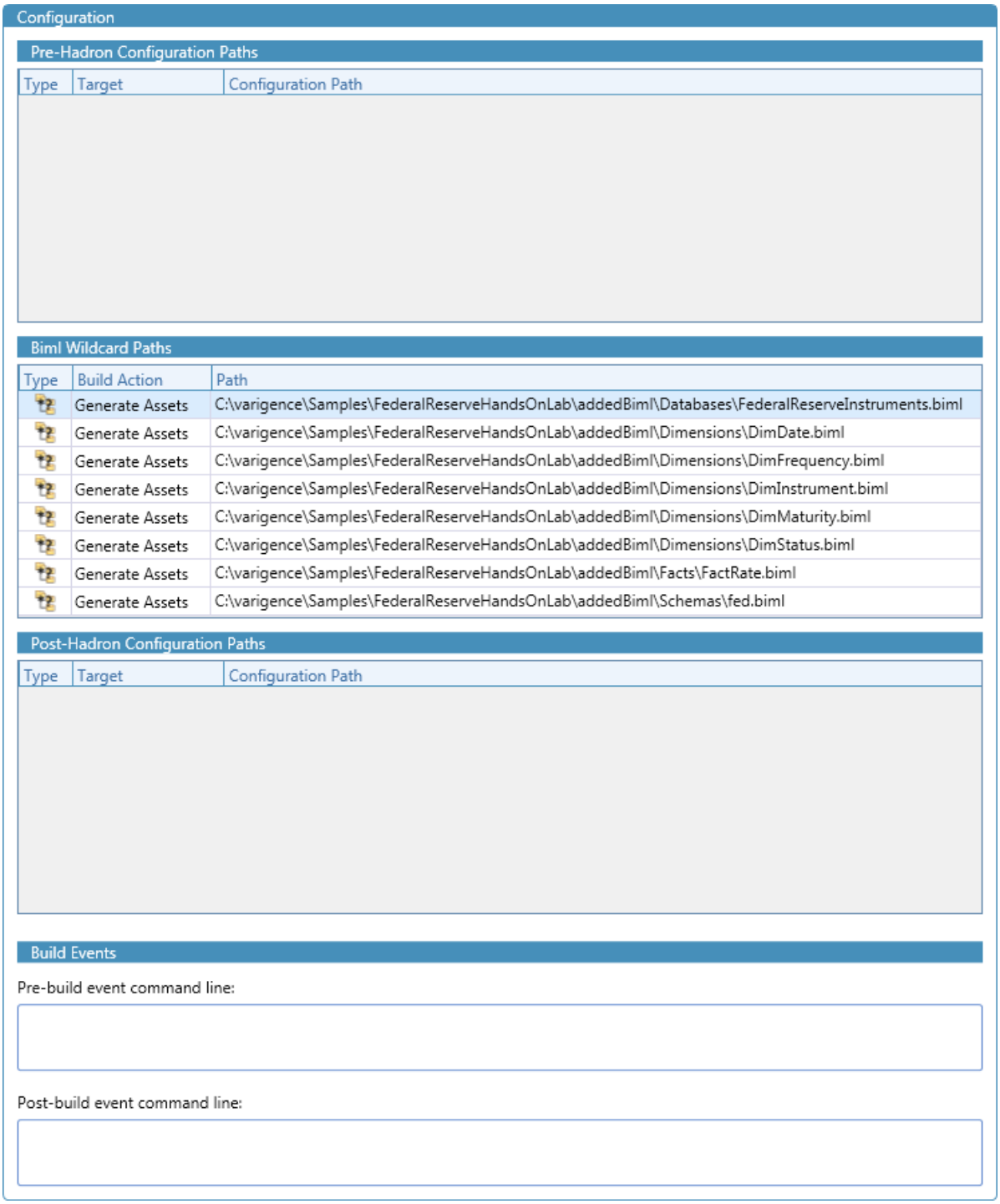
Afterwards, four new files will be added to Project View:



These are the newly generated configuration files. Double clicking on a configuration file opens it in the configuration designer.

The Configuration Designer

The configuration designer, displaying the ddlConfig.bimlproj file's contents, consists of 3 data grids and 2 text boxes.



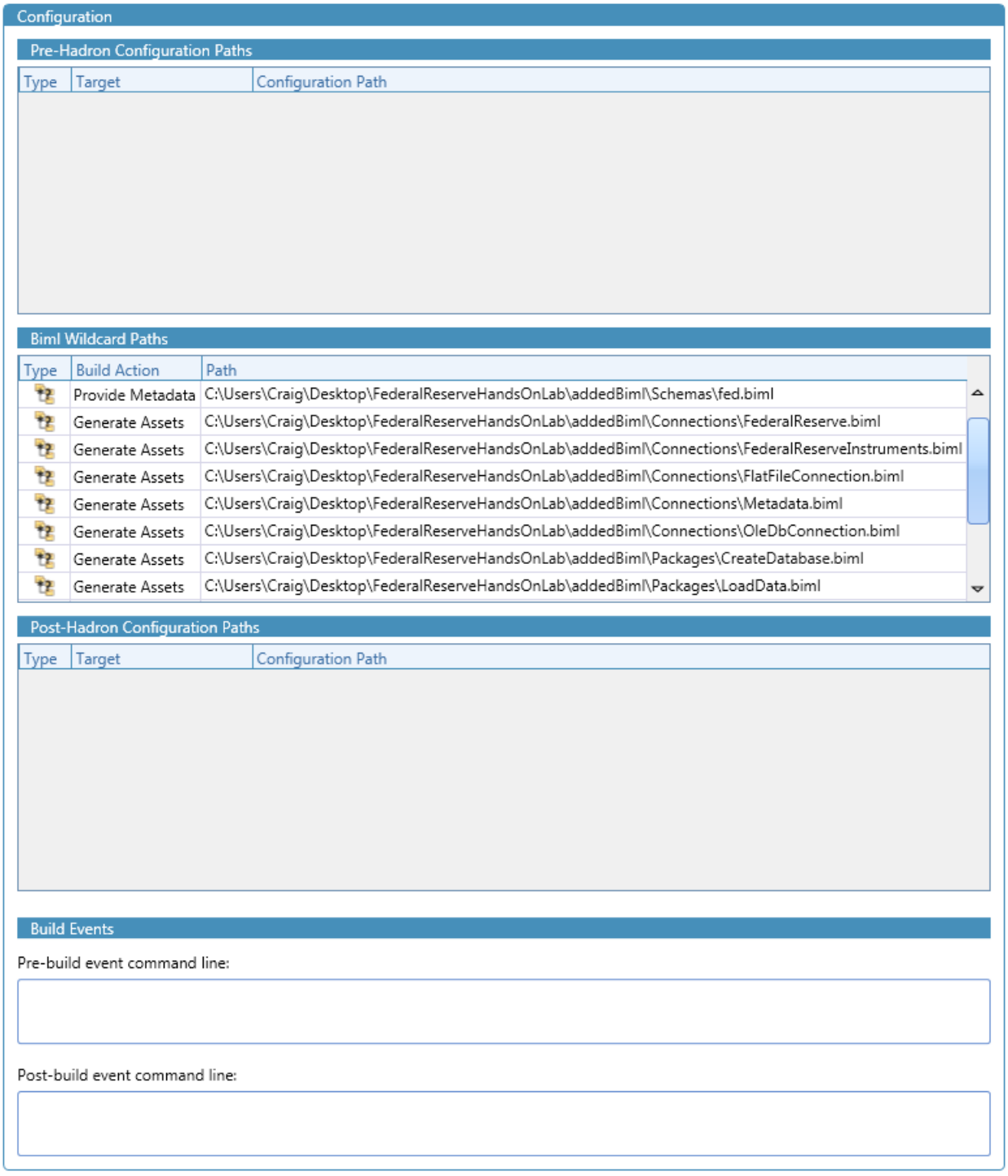
Biml Wildcard Paths

Notice the Biml Wildcard Paths data grid in the center. In the configuration designer, each Biml file path is listed in its own row in this data grid.

To the left of each file path is an Build Action column. Each cell in the column provides a combo box that lets you choose Generate Assets or Provide Metadata for the Biml file's Build Action. These descriptions map directly to the Emittables and Includes groups from past Mist versions.

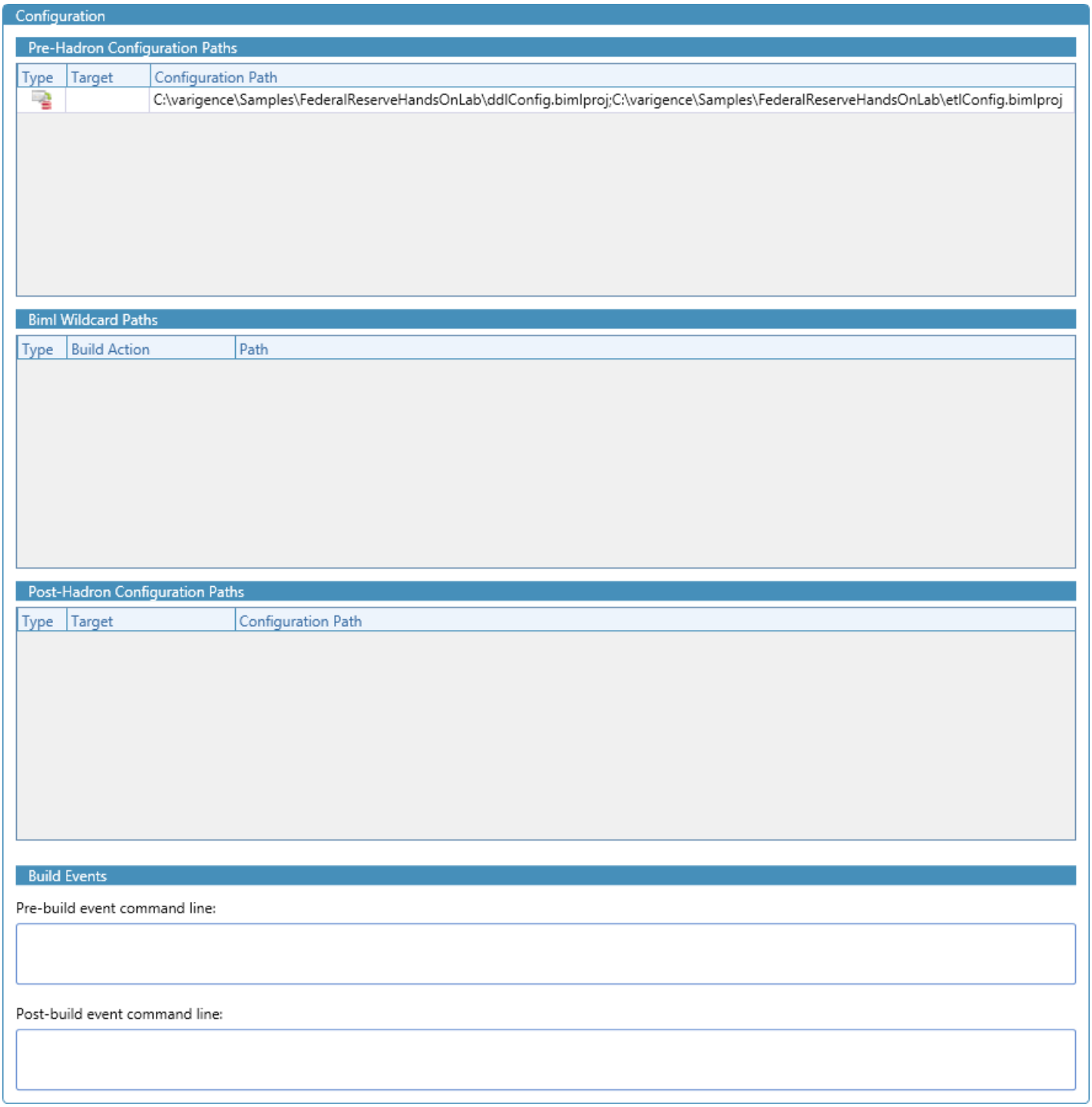
Remember that the listed Biml files contain the yellow and blue highlighted assets from the sample project's logical view. As explained, this configuration includes Biml files that contain DDL assets.

Opening the etlConfig.bimlproj in the configuration designer shows the following:



This configuration lists Biml files that include ETLs and related assets, which are the green and blue highlighted assets in the sample project's logical view.

Next, this is the configuration within the buildConfig.bimlproj file:



First, notice that the Biml Wildcard Paths data grid is empty. That's because this configuration doesn't directly process any Biml files. Instead, it references other configuration files.

Pre & Post Hadron Configuration Paths

Referencing other configuration files enables users to refactor their build process. In this example, the build configuration references the dllConfig.bimlproj and etlConfig.bimlproj files in the sole row in the data grid. If a user builds their project with this configuration, the build engine will examine this configuration and see that it references two other files. Then, it will open ddlConfig.bimlproj and attempt to build all the Biml files listed within it. Next, it will open etlConfig.bimlproj and attempt to build the Biml files it lists.

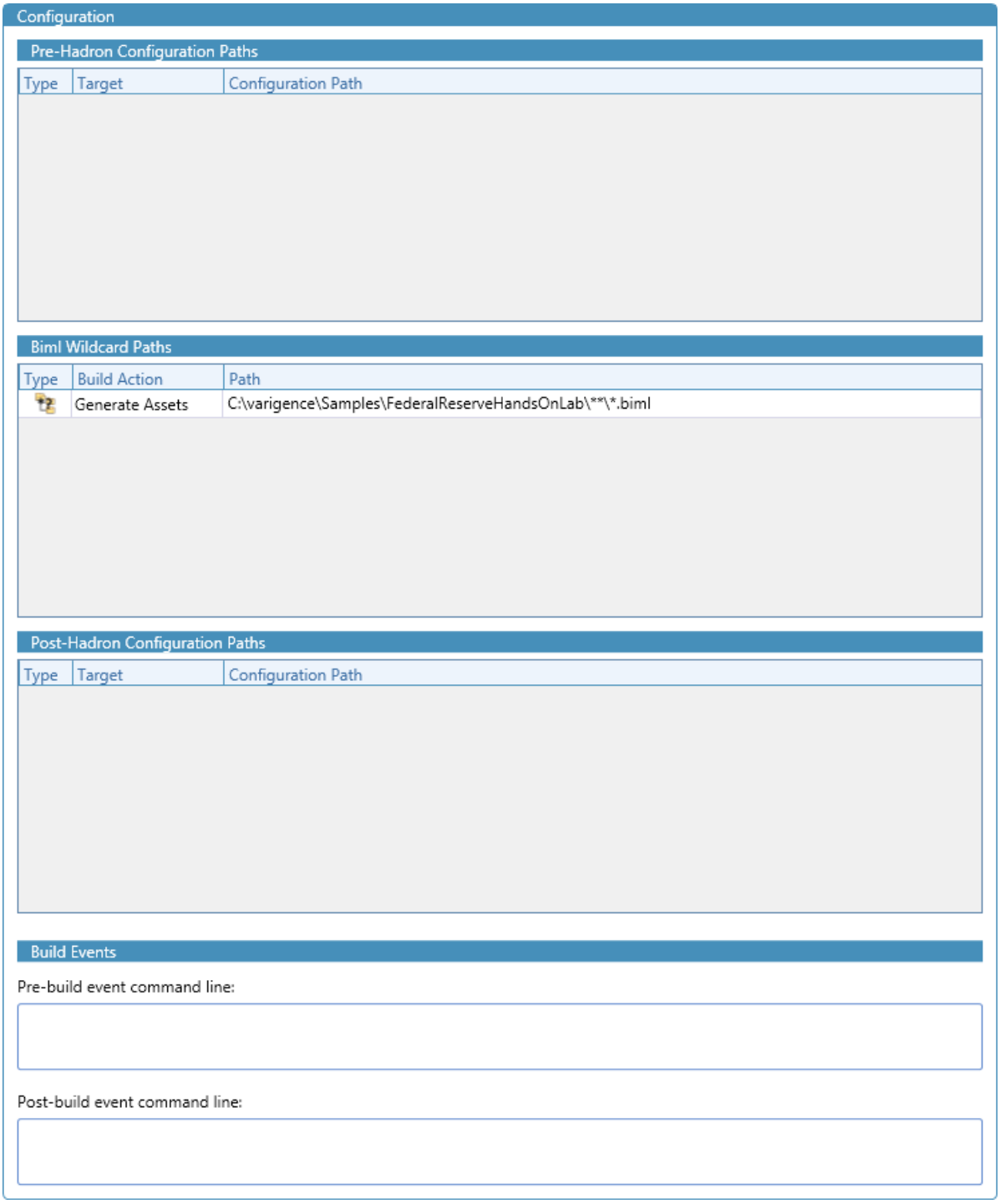
The semicolon that separates the two configuration paths is important syntax. For configurations paths and Biml wildcard paths, multiple file paths can be listed in the same row by separating them with a semicolon. The paths are processed from left to right.

Notice that the row's Target cell is empty. The Target property is a MSBuild specific property and is discussed later in this article. It's typically not needed and can remain empty.

Note that configuration paths and Biml wildcard paths are not mutually exclusive; a configuration can contain both. In this case, the

configuration paths in the Pre-Hadron data grid are processed before any of the files in the Biml wildcard paths data grid are built. The configuration paths in the Post-Hadron data grid are processed after the Biml wildcard paths.

Finally, the configuration within the allConfig.bimlproj file is as follows:



Wildcards

As aforementioned, this configuration file contains all Biml files under the project directory and its subfolders. To accomplish this, two pieces of MSBuild syntax are used. The first is the asterisk character, used in *.biml. The asterisk means to add all files with the biml file extension. The double asterisk character indicates that MSBuild should look for Biml files in the FederalReserveHandsOnLab directory and subdirectories.

The asterisk and double asterisk characters are examples of wildcards. Wildcards are characters that can specify a group of items in an abstract way. Both Biml file paths and configuration paths can use wildcards.

To see additional examples of MSBuild syntax for specifying files, review this article.

Pre and Post Build Event Command Lines

The configuration designer also provides convenient access to MSBuild's build events. A build event is a command that MSBuild performs at a particular stage in the build process. The pre-build event occurs before the Mist build begins and the post-build event occurs after a build successfully ends.

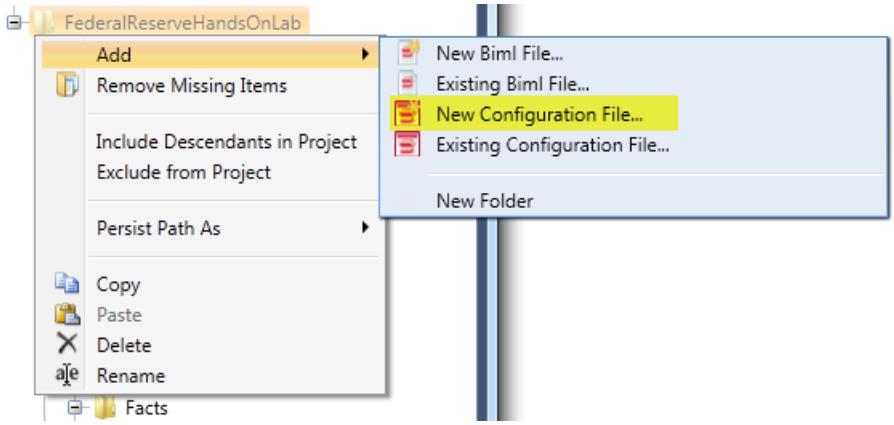
The configuration designer allows users to enter build events via the Pre and Post build event command line text boxes. Each build event should be entered on its own line. The commands that can be executed include any commands that can be run from a Windows command prompt. Bear in mind that commands may depend on the Path environment variable to work correctly.

Creating Custom Configurations

Default configurations can be useful in numerous scenarios, especially when serving as a starting point for a custom configuration. However, there are times when a user will want to start from scratch.

Creating configurations

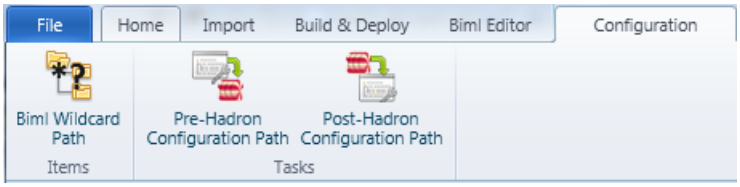
To add a new configuration to your project, right click on any folder in Project View and select Add -> New Configuration File.



Use the New Configuration File dialog to name your configuration file and choose its location. Once done, press Save to create the file and add it to Project View.

Editing configurations

Interacting with the configuration designer is very easy. The Configuration ribbon contains buttons for adding Biml Wildcard Paths, Pre-Hadron Configuration Paths, and Post-Hadron Configuration Paths.



Additionally, you can drag and drop a Biml file from Project View onto the Biml Wildcard Paths data grid to create a new row with the Biml file's path in it. Similarly, you can drag and drop a configuration file from Project View to the Pre and Post Hadron Configuration Paths data grids to add a new row containing a path to the dropped configuration file.

Rows in these data grids can be deleted by selecting the row and pressing the delete key.

The configuration paths and Biml file paths can be edited by double clicking on the cell. You can use the cell's ellipses button to open a file dialog, for getting a file path.

MSBuild Specifics

XML Editor

Along with the Configuration designer, Mist provides a second Configuration tab that holds a XML editor. The XML editor gives users

complete power over a build configuration, leveraging all the capabilities of MSBuild.

The configuration designer and its XML editor are synchronized. Any changes within the configuration file's autogenerated PropertyGroups and ItemGroups will overwrite values in the configuration designer. Likewise, any changes in the configuration designer will overwrite values within the configuration file's autogenerated PropertyGroups and ItemGroups.

This configuration file references the ddlConfig.bimlproj file by using a PreConfiguration item. When this configuration is built, MSBuild will execute the ddlConfig.bimlproj file, using a MSBuild task. The PreConfiguration item's Targets metadata maps to the Target cell in the Pre-Hadron Configuration data grid. More information on MSBuild targets can be found here.

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="3.5" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(AssemblyPath)\Varigence.Biml.targets" Condition="true" />
  <ItemGroup Condition="true">
    <PreConfiguration Include="C:\varigence\Samples\FederalReserveHandsOnLab\ddlConfig.bimlproj">
      <Targets>
      </Targets>
    </PreConfiguration>
  </ItemGroup>
  <!--
  Any changes outside of the auto-generated ItemGroup and PropertyGroup tags
  will be ignored by the Configuration designer.
  Any content within the auto-generated ItemGroup and PropertyGroup tags can be overwritten by the Configuration designer.
  -->
</Project>
```

This configuration file lists paths to the Biml files that will be compiled when building. The Sources element is for files whose assets should be generated when building; the element maps to the Generate Assets build action in the Configuration designer. The Includes element is for files whose assets should only be referenced, not generated, during compilation. This element maps to the Provide Metadata build action.

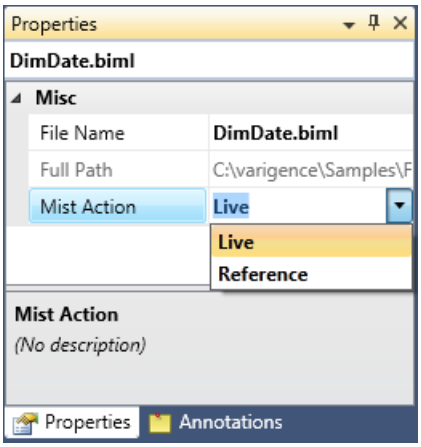
```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="3.5" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(AssemblyPath)\Varigence.Biml.targets" Condition="true" />
  <ItemGroup Condition="true">
    <Includes Include="C:\varigence\Samples\FederalReserveHandsOnLab\addedBiml\Dimensions\DimDate.biml" />
    <Sources Include="C:\varigence\Samples\FederalReserveHandsOnLab\addedBiml\Databases\FederalReserveInstruments.biml" />
  </ItemGroup>
  <!--
  Any changes outside of the auto-generated ItemGroup and PropertyGroup tags
  will be ignored by the Configuration designer.
  Any content within the auto-generated ItemGroup and PropertyGroup tags can be overwritten by the Configuration designer.
  -->
</Project>
```

Using the Project View Configuration

Although configuration files are useful and offer powerful customization options, there are times when they're unnecessary. Thus, Mist allows you to build your project without relying on configuration files.

All Mist projects have a built-in "Project View" configuration. This configuration is the top item listed in the configurations dropdown in the Build & Deploy ribbon tab. When building with the Project View configuration, Mist automatically targets all Biml files included in your project.

To control a Biml file's Mist Action, select the Biml file in Project View and then display the Property Grid tool window. Each Biml file has a Mist Action property, with values of Live and Reference.



Biml files that have a Live mist action are treated as source files. Biml files that have a Reference mist action are treated as metadata files. Remember that you can use multi-select (via Ctrl+Click or Select+Click) to select multiple Biml files and set their Mist Actions simultaneously.

Project View Idiosyncracies

Mist's Project View reflects the state of folders and files on disk. From within it, users can:

- Add, delete, rename, copy, and move files and folders
- Include and exclude files and folders from the project
- Interact with Subversion or TFS
- Open files in their associated editors
- Build individual files

For those acquainted with Visual Studio or other IDEs, Mist's Project View will feel very familiar. However, Mist's Project View has some unique characteristics that this article explains.

Root Folders

Project View allows a user to view, and interact with, their file system. However, Mist doesn't display the entire file system in Project View as it could be overwhelming, and a user likely wants to see only a small portion of it.

Thus, root folders are used to control what portions of the file system are displayed within Project View. When creating or opening a Mist project, at least one root folder will typically be present. Within Project View, it's possible to navigate the file system beneath a root folder.

If a user wants to interact with other parts of the file system, he can add additional root folders by right clicking on the root project in Project View and selecting the Add Root Folder command.

Note that adding a folder, as a root folder, has no impact on the file system.

Show All Files

When a new root folder is added, its contents can be viewed in Project View in one of two ways. The first option is to enable the Show All Files feature. This feature allows you to see all folders and files on disk that are underneath the project's root folder, even if they're not included in the Mist project. A grayed out folder or file icon indicates that the item is excluded from the project.

Include Descendants in Project

The second option is to right click on the root folder and select Include Descendants in Project. This command includes all folders and files, underneath the root folder, in the project. This isn't done automatically since a user may want to only include specific files under a root folder. Furthermore, for very deep directory structures, this operation can produce a noticeable delay.

Persist Path As

When opening a Mist project, Mist uses a root folder's path to locate included files beneath the root folder. Thus, Mist is dependent on the project file containing accurate root folder path information to load a project's files.

When sending a Mist project to another user, it's possible that the root folder paths on the sender's machine won't exist on the receiver's machine. As a result, the receiver won't be able to view the project's files within Mist.

To handle this scenario, Mist allows users to store their root folder paths in absolute or relative form via the Persist Path As setting. This setting is exposed in a root folder's context menu and offers two choices: Absolute and Relative.

Absolute indicates that the complete root folder path is stored in the project file, starting at the drive letter. If the project is opened on another machine, the same root folder needs to exist at the same location. Otherwise, the root folder's files won't be found.

Relative means that the root folder's path is stored relative to the project file. Thus, the project's files will be located as long as the directory structure, that contains the project file, is preserved when sending the Mist project to other users.

Therefore, to send a project to someone, a sender should typically set a root folder's **Persist Path As** setting to relative, save the project to preserve the setting, and then send the user the entire directory tree that the project uses. This guarantees that the receiver can successfully use the project.

Setting **Persist Path As** to relative is also useful when multiple users share the same files, perhaps via source control, but their working folders are in different locations. As long as all users place the project file in the same location, relative to the project's other files, the project will load correctly when it's opened.

A scenario where **Persist Path As** should be set to Absolute is when the root folder points to a location that both the sender and receiver can access, such as a network location. In this case, the sender won't need to copy the network location's contents with the project because the receiver can access it using the same path.

Refresh

Although Mist's Project View displays the state of folders and files on disk, it will not automatically update itself when changes happen outside of Mist. For instance, Project View may be showing a project's Connections directory that contains three Biml files. Outside of Mist, however, a user copies two additional Biml files into the Connections directory. When returning to Mist, Project View will still only show three files in the Connections directory.

To update the Project View, so it reflects the current state of your file system, use the Refresh button.

Moving Files in Subversion

Mist's Project View has integrated support for common Subversion and TFS commands. However, it's important to note that when using Project View to move files in Subversion, the moved files can be orphaned from the repository. The underlying issue is that Project View performs file moves within Explorer, but Tortoise fails to recognize file moves within Explorer. Thus, Tortoise treats the moved file as missing from its original location and ignores the moved file at its new location. The workaround is to manually add moved files back into Subversion.

BimlStudio User Guide

The BimlStudio User Guide describes all aspects of the BimlStudio user interface.

The BimlStudio User Guide is broken up into three parts. The first part provides a step by step walkthrough for building assets for use in a sample cube, leveraging the AdventureWorksLT database. The second part describes each BimlStudio editor, tool window, and ribbon button. The third part includes miscellaneous topics that don't fit into the first two groups.

BimlStudio Forum

If you have questions that aren't addressed in this guide, you can search and post in the BimlStudio forum for help.

Email Support

While the forums are the best way to get answers, if your question contains proprietary, or other, information that you would prefer not to post publicly, private email support is available at support@varigence.com

BimlStudio Documentations

BimlStudio User Guide

The BimlStudio User Guide describes all aspects of the BimlStudio user interface.

The BimlStudio User Guide is broken up into three parts. The first part provides a step by step walkthrough for building assets for use in a sample cube, leveraging the AdventureWorksLT database. The second part describes each BimlStudio editor, tool window, and ribbon button. The third part includes miscellaneous topics that don't fit into the first two groups.

BimlStudio Forum

If you have questions that aren't addressed in this guide, you can search and post in the BimlStudio forum for help.

Email Support

While the forums are the best way to get answers, if your question contains proprietary, or other, information that you would prefer not to post publicly, private email support is available at support@varigence.com

BimlStudio 2018.1 Release Notes

Significant changes between BimlStudio 2017 and BimlStudio 2018.1

Azure Data Factory Support

Added support for all ADF version 2 items and is current as of 3/20/2018. This includes the ability to:

- Create all new V2 Linked Services
- Create Activities
    - Create Copy Activities Create any supported V2 Sources Create any supported V2 Sinks
    - Create Transform Activities
    - Create Control Flow Activities
- Create Triggers

Other

- Updated expression grammar to better handle whitespace in expressions and to improve the parse tree

BimlStudio UI Updates

- Improved styling in the Package Import dialog, by widening the file path textbox, adjusting the height of the file path textbox, and making the buttons uniform. Also eliminated unnecessary columns in the DTSX file path grid.
- Improved styling for Project Settings editor, by widening the buttons and increasing the height. Also applied the styling to be consistent with the rest of BimlStudio.
- Fixed button width on trial key dialog.
- Changed all "2017" references to "2018".
- In Package Import dialog, we now provide users the option to create an SSIS Import Options Json file, in order to save your SSIS import options for use with command line imports.
- Added support for guaranteeing that a Biml file's expansion result hasn't changed since the previous expansion.
- Revamped the Debug utility in BimlFlex projects, now users are able to extract their metadata, while also zipping up their project directory, for aid in debugging through Varigence support.
- Changed "recent path" saving to help with more consistent saving of recent paths.

SSIS

- Added a checkbox to Import Packages dialog box to allow users to hide package creation metadata in the imported package.
- Changed `AutoAdjustBufferSize` to `autoAdjustBufferSize` during SSIS 2016 emission.
- Fixed IsPac File emission to work correctly.
- Fixed bug where sensitive parameters were not getting saved to the Params file.
- Fixed bug where sensitive parameters were not getting encrypted for OleDbConnections.
- Added support for SSIS 2017.
- Updated to support and build Script Components for SSIS 2017.
- Fixed issue of missing default values that broke SSIS emission for some components such as CDC Splitter, which would fail to load in SSIS when some default values were supplied.
- In imported packages, the tasks are ordered in a more logical and readable fashion that matches what the user would see in SSDT.
- In imported packages, the contents of Data Flow tasks are organized in a more logical and readable fashion, one that follows the flow of information through the pipeline.
- Exposed VersionGUID property on packages, so that user can specify a GUID, if it is not specified, then it is randomly generated as before.

BimlStudio Bugs

- Fixed issue with extra `xmlns` declarations when expanding BimlScript files in BimlStudio.
- Fixed possible exception in drag and drop in BimlStudio.
- Fixed the uninstall survey link in the bootstrapper to point to the correct address (www.varigence.com).
- Fixed issue where BimlStudio's "Format Document" functionality disrupted Annotation Tags with variable values.
- Fix for Null Reference Exception in `BimlTemplateLoader`.
- Fix for Biml files being parsed while unloaded for editing.
- Fixed issue where changes to the MsBuild paths, BimlEngine paths, and DiffViewer settings were not being tracked by BimlStudio.
- Fixed issue with new BimlFlex projects where 32 bit versions of BimlStudio were not finding the 64 bit BimlFlex.xlsx (for use with 64 bit Excel).

Source Control

- Fixed bug where folders under source control were displaying as "checked-in", even when they contained files with pending changes.
- In the History tab, the number of changes was always being displayed as 1, now it accurately displays the correct number of changes that were pushed in a commit.
- More descriptive label for renames that displays the changed portion of the file path.
- Disabled the UI when committing to source control, in order to prevent undesired behavior.
- When launching the file comparison tool in the History tab, the commit history list box is now disabled, as changing your selection while the tool launched produced unpredictable results.
- In Project view context menus, disabled "Revert" command for newly added files.
- When launching the file comparison tool in the History tab, it is now done asynchronously, as we sometimes need to get information from a remote server, which caused the UI to lock.
- Fixed bug where new projects without a template were not being tracked as eligible for source control.
- For Svn, previously, in order to see the historical file contents, a version of the file had to still exist locally, now if the file has been completely deleted or renamed, we are still able to retrieve its contents from the server.
- For Git, fixed bug where commits were getting added to both "Outgoing" and "History", causing duplicates in the History tab.
- For Git, fixed bug that was preventing file comparisons.
- For Git, fixed bug where the push/sync commands would still fire after a failed fetch command.
- For TFS, Fixed bug where BimlStudio was not recognizing TFS projects that had a space in the file path.
- For TFS, improved error logging to help with diagnosis of issues.

Tabular Support

- In `AstTabularRelationshipNode`, added a `SourceColumn` property with a schema binding of `TabularColumnName`.
- In `AstTabularRelationshipNode`, `TargetColumn` property now has a schema binding of `TargetTabularColumnName`.
- In `AstTabularModelNode`, `OutputModelLabel` property is changed to `OutputLabel`. Also added a `Relationships` property composed of a collection of `AstTabularRelationshipNodes`.
- Added ability to specify tabular column relationships at the model level, in addition to within the `AnalysisMetadata` section of a table.
- Now displays the correct names of the various assets in Tabular relationships and perspectives.
- Added documentation to the following Tabular enumerations: `InferRelationshipMode`, `TabularModelPermission`, `TabularRelationshipCardinality`, `TabularRelationshipFilterDirection`.
- Changed friendly type name from `TabularTableColumn` to `TabularColumn`
- Now setting correct values for `DeploymentServerDatabase` and `DeploymentServerCubeName` in the `smproj` file in all instances
- Now setting the name for roles correctly in all instances
- Fixed issue where underscores in asset names would not be displayed in the `CubeProject` and `PackageProject` designers

- Added descriptions to emitted SSAS Tabular roles based on Biml principal annotations
- Added DisplayFolder property to SSAS Tabular Table Columns

Performance

- Fixed issue where an ICE would be generated on builds with warnings.
- Added the `tempPath` setting to the Biml compiler. This setting specifies the temporary path that will be used for intermediate assets during the build process.
- Removed memory leak due to repeated `xslt` transforms.
- Now auto-generating BimlC Response file on every build and placing it in the project directory.
- Reduced memory and eliminated Out Of Memory exceptions when creating ISPAC files for large SSIS projects.

Provider Support

- Fixed data type mappings for Excel `GetDatabaseSchema()`.
- Added logging to `GetQuerySchema()` method.
- Added compression to SQL Server `GetDatabaseSchema()`.
- Change the DB2 provider to reference `SYSIBM` system tables instead of the `SYSCAT` views, as they are sometimes missing.
- Fixed Excel support in OLEDB connections in Biml.
- Fixed date datatype mapping in ACE OLEDB provider.
- Added custom type mappings to `ExternalDataAccess.GetQuerySchema()`.
- Fixed OleDb source and destination for `geography`, `geometry`, `hierarchyid`, and `image` data types.
- Fixed data type mappings in Flat File Source around `NTEXT` versus `nvarchar`.
- Fixed data type mappings in CDC Source.
- Fixed issue where ACE OLEDB provider was not being used when the connection string called for it.

Biml Language

- Added ColumnStore and ColumnStoreArchive to `TableCompressionType` enumeration.
- Fixed spelling errors in Biml documentation for `ProcessionOrder.Sequential`, `ProcessingOrder.Parallel`, and `ContainerConstraintMode.LinearOnSuccess`.
- Added correct schema binding for `TextQualifier` to fix the misspelling in the Biml language.
- Implemented `ICollection` interface for `TierFilteredVulcanCollection` to fix issue with Preview when using extension methods from `codefile`.
- Eliminated unnecessary and potentially buggy code for finding schema names of items based on being referenced rather than definitions, resulting in items unexpectedly emitting as `<AstNode>` in XML.
- Fixed Xml emitter for singleton references in Biml with multiple schema bindings, for example, in `AstLookupNodes`, all connection references were emitted as `OleDbConnectionName`, and now they are accurately emitted as either `OleDbConnectionName`, `CacheConnectionName`, or `CustomSSISConnectionName`.
- Improved Json emission
- Fixed `EncryptSensitiveWithUserKey` emission issue

Bundles

- Fixes for bundle toggle files settings and trial licensing
- Changed the name of `Visibility` bundle property to `File Entry Visibility` in order to avoid name conflict warning.
- Fixed issue where bundle settings overrides to toggled-off files would not always work.
- Added fix to clear bundle file cache when removing a bundle.
- Performance improvements for bundles to prevent recompilation of files that are guaranteed to not have changed.
- Fixed an issue with extension points that take dynamic objects using the wrong assembly version of those dynamic objects.
- Removed message about script components from the Setup Dialog.
- Small fix to clear bundle file cache when removing a bundle.

Project Settings and Command Line Options (CL)

- Updated license key specification on command line to use the entered license key, even if it is not successfully saved to the registry. This will help in CL and automation scenarios where proxy accounts are used
- Ability to create `Bimlc` response files for command line compiling
- Added DDL Build mode options to project settings and CL options for selecting the type of DDL script emission to do for relational objects in the project
- Added an option in the Project Settings panel to create a `Bimlc` response file for use with `bimlc.exe`
- Set `IsCleanOutputFolderEnabled` to true by default for new projects
- Fix for bundle settings files not loading correctly when overridden on the command line
- Added `DdlBuildMode` property to Build Settings
- Fixed Bundle Settings on command line and MSBuild
- Fixed issue where documentation templates passed via the command line were not being used in the build process
- Fixed issue where 64 bit BimlStudio was not finding existing 32 bit BimlEngine dll, and vice versa

BimlScript

- Fixed autocomplete bug with BimlScript `global/applytocallbimlscript` directive attribute, `active` was showing up twice in the autocompletion list, and `applytocallbimlscript` was not showing up at all
- Added caching timestamps so that extension points and CallBimlScripts that use dynamic objects can still be cached when neither the extension point nor the dynamic object schema has changed
- Fixed bug where passing in the parent item as null in a BimlScript object constructor was generating the wrong XML
- Added `Varigence.Biml.Flex` to the list of default namespaces for BimlScript code
- Fix to make global include files work with reference Biml files
- Updates to BimlDoc templates to have them align with what is used in the emitter defaults
- Small fix for intellisense in extension languages

Miscellaneous

- Added PDF version of the EULA

Significant changes between Mist 4.0 Update 1 and BimlStudio 2017

Breaking Changes

- Moved PathAnnotation from PrecedenceConstraints to TaskflowInputPath.
- Clean output folder is now set to false by default in AstEngine's compiler settings.
- Fixed issue in CallBimlScript property directives where all properties were required and marking them as required would create an error. To fix this, we apply all of the required properties first and then the optional properties in order of specification. Since required properties basically didn't work before, this shouldn't affect any actual code, but it should be called out as a potentially significant behavioral change.

BimlStudio Rebranding

- Updated version to BimlStudio 2017 v5.0.
- Updated splash screens, logos, and color scheme to match Biml color scheme.
- Removed all instances of the text "Mist" from the product and all code. If you were using any unsupported Mist APIs, replace all instances of "Mist" with "BimlStudio".
- Updated BimlStudio version to 5.0 in the Registry.

Major Features

- Biml global includes
- Biml Bundles
- New source control UI including Git, Visual Studio Online, and Team Explorer Everywhere support
- Significant performance improvements
- New installer
- SSIS 2016 support
- SSAS Tabular Emission
- Usability enhancements throughout the UI
- Added multi-product support to license keys and better offline support

Platform

- Moved all DLLs and executables to use .NET 4.6 rather than .NET 3.5.
- Removed warnings from a couple of framework transformers.
- BimlStudio is now truly x86 or x64. This shouldn't be visible to end-users, but it was necessary to begin hosting HTML/JS/CSS controls natively in BimlStudio. The command line compiler is unaffected and remains AnyCPU.
- Updated EULA to cover frameworks including BimlFlex.

Biml Bundles

- User Interface
- Bundles are now shown in project and logical views.
- Bundle designer added.
- Wired up bundle settings editor.
- Biml Language
- Added the <#@ extension #> directive to customize BimlBundle extension points.
- Biml API
- Added extension and bundle related Call methods to FlowTextTransformation.
- Biml Code Editor
- Added intelliprompt for extension bundles and extension points. targets still need to be done.
- Added ability to specify templates to a command line build to support extension discovery.

- Added support for passing template files into builds so that bundles can use them to find extensions, among other benefits.
- Added support for passing bundles into builds.
- Added support for specifying that a framework or a transformer lives inside a bundle.
- Added initial support for bundle overrides.
- Added msbuild and command line inputs for BimlFrameworkBundleFilePath,TransformationScriptBundle, Bundles, BundleOverrides, and Templates.
- Fixed crashing issue when PluginDirectory doesn't exist. This only manifested in BimlOnline.
- Updated PathResolver to work with Bundles. There is some more stabilization and testing to do - especially around *BUNDLE* paths.
- Added support for Bundle setting management and serialization.
- Manifests
- Added icon support for EP categories in bundles.
- Added new file template support in bundles for Eps.
- Added the ability to split manifest files in two, including a common file and multiple variants for application specific bundle modes.
- Fixed an error in the new BimlFlexExcel MetadataInstance.biml file.
- Added MetadataInstanceName and MetadataModelName as properties inside of the bundle manifest.

BimlFlex

- Added Extension Point information to BimlFlex ribbon.
- Wired up icons for BimlFlex ribbon.
- Ribbon updates for SetupBimlFlex and Generate Scripts.

Performance

- Added lazy loading of collections to BimlStudio view models. This will significantly reduce memory consumption for large projects.
- Improved performance of CallBimlScript by around 30X by adding caching when possible.
- Added caching of internal build information for ScriptTask Projects and Script Component Projects. This will massively increase build performance where the same script task or component is reused in many locations in a project (provided that the ProjectCoreName is also reused).
- Added caching to dynamically generated metadata model class, which provides massive performance improvement for RelatedItem interrogation in large metadata instances.
- Memory improvements when closing documents.
- Improved performance issue in the editor when there were large blocks of plain text.

Logical View

- Fixed issue where generating the DropAndCreateDDL from BimlStudio context menu would crash Mist if the table had errors.
- Added persistence of logical view expansion state between invocations of BimlStudio.
- Added a new Bundle Extensions group to the Library of the Logical View.
- Fixed bugs with logical display folder calculations when files can jump between multiple filtered folders (Utilities, Transformers, Extensions).
- Added delete confirmation dialogs for measure groups and dimensions in the logical view.
- Adding "Delete Unscripted" to top-level Logical View nodes.

Biml Code Editor

- Fixed issue where opening a standalone Biml file could crash Mist.
- Fixed quick info and code completion in standalone Biml files.
- Improve selection highlighting syles in BimlEditor to make it easier to read.
- Fixed intelliprompt in include scenarios.

- Fixed intelliprompt in multi-line text nuggets.
- Made some additional properties Advanced to prevent them from showing up in intelliprompt completion lists.

All Text Editors

- Fixing issue where BimlStudio would open txt and csv files in an external editor when opened from the logical or project view.
- Added Cntrl+Shift+H key binding for Replace All.
- Added support for find/replace all to all text file types.
- Added support to find/replace all to work with multiple root folders.
- Added line/col status bar support for all text file types.
- Added a dialog that shows the number of occurrences replaced in a ReplaceAll operation.

Other BimlStudio UI

- To enable FIPS compliance, changed all internally used MD5 hashes to SHA1 hashes.
- Fixed sizing of the project converstion dialog box.
- Fixed an issue where Unicode characters were not being properly displayed in output window for MSBuild executions.
- Added the friendly type name of the candidate object in all multi-select confirmation dialogs (transformer and table import overwrite confirmation dialogs).
- Fixed issue where several ribbon buttons would not be disabled when a project is not loaded.
- Changed settings and recent files to save more frequently.
- Adding bundle manifest files as a supported BimlStudio miscellaneous text file type.
- Added support for Bundle settings and a settings editor in BundleEditor tab.
- Force Transformers, Globals, and Extensions to never be Live.

Biml Language

- Added Biml language support for PermittedValues in AstMetadataModelPropertyNode.
- Added support for global includes in CallBimlScript target files.
- New global directive to disable the global include for CallBimlScript files, called ApplyToCallBimlScript
- Added LinearOnCompletion and LinearOnSuccess constraint modes to packages
- Update to allow project and package parameters to be used whereever variables are in SSIS.
- Modified ScriptComponent logic so that BlobColumns are created when needed rather than using String columns.
- Added mapping output from TextTemplating engine to permit line mappings from expanded BimlScripts to the source BimlScripts.
- Added the DisablePackageConfigurations property to packages. This is primarily for cases where imported packages reference package configurations but have them disabled.
- Added SsisDataTypeOverride functionality.
- Set default value for LogicalDisplay folder to empty string rather than null. This simplifies coding when LogialDisplayFolder is used as a lightweight intermediate metadata source.
- Changed BimlScriptDirectiveLanguage to include extensionpoint directive tokens.
- Added TargetSeverVersion to SqlServerDqsConnection for Ssis2016 and above.
- Added logging support to the DqsCleansing component.
- Made everything related to script projects into scope boundaries.
- Metadata Models
- Added permitted values collections to metadata models.
- Added IsUiEditorVisible to Metadata Model Properties.
- Added support to metadata model relationships for specifying filtered lookups in the UI.
- Entity IsUiEditorVisible

Biml API

- Added ConvertExcelDateToDateTime method to DataTypeConversion.

- Changed all overloads of CallBimlScript* that product a dynamic object output to be called CallBimlScript*WithOutput. This is to avoid issues with VB not handling output parameters correctly.
- Added IsBackgroundCompilation property that can be accessed from BimlScripts. It lets you conditionalize things like popup dialogs or file writes to happen only when you're doing a real build or a manual execution.
- Added an extension method for easier access to connection string values.
- Added support for CustomOutput object in CallBimlScript and its relatives.
- Added CallBimlScriptContent to FlowTextTransformation so that BimlScript literals can easily be called.
- Added MakeSsisSafe extension method for strings.
- Fixed a logic error in AstTableNode.GetColumnComparisonList extension method when emitNullComparison is set to true.
- Fixed an issue where RootNode.GetTag() would not work on file updates within live BimlScripts.
- Added an AstDestinationTransformationNode abstract class - this is primarily useful for transformer targeting.
- Added MetadataModels, MetadataInstances, and OfflineSchemas properties to AstRootNode and TierFilteredAstRootNode.
- Fix for issue where GetBiml would not include some changes to collections that had been made programmatically.
- Added flat file extension method for generating table columns from flat file columns.
- Fixed issue where multi-type generics could not be used as arguments in CallBimlScript.
- Fixed issue where CallBimlScript files were not working with ObjectTag in Live BimlScript scenarios.
- Metadata Models
- Fixed some issues with metadata model wrapper where property names and parent names were not being written to the validation items correctly.
- Added instance information to dynamic objects for metadata validators.

Database Providers and Type Mappings

- Added formal support for the DevArt MySql provider.
- Added formal support for Ingres ODBC and ADO.NET.
- Added formal support for SAS local data provider (e.g. sas7bdat files).
- Fixed issue where custom types from SQL Server were not being imported correctly.

SSIS Components

- Added Biml elements for Azure Feature Pack, including:
- Task details controls and images
- Models for connections and tasks
- SSIS emitters and importers

Source Control

- Added Git source control
- Added support for Team Explorer Everywhere
- Added support for Visual Studio Online source control authentication
- Moved source control objects to BimlStudioExtensions so that they can be open sourced in future
- Complete overhaul of source control UI

Visual Basic

- Updated BimlEngine so that dynamic objects will work with both VB and C#.
- Added VB code for metadata model dynamic objects.
- Removed default timeOut arguments for all of the external data access APIs and replaced them with explicit method overloads. This is primarily to better support VB, which does not handle overloads with default arguments as elegantly as C#.

SSIS Import

- Fixed an issue in the SSIS importer where an empty string ForcedExecutionValue might be set.
- Added support for a status object when import is run programmatically.

- Fixed issue where SSIS importer would not correctly detect the SSIS version in some cases.
- Fixed SSIS 2008R2 package emission issues.
- Fixed bugs with emitting SSIS annotations.
- Preventing crash in importer when component files cannot be read from inside of a ScriptComponent.

Table Import

- Fixed issue where identity information was not being imported properly from SQL Server tables.
- Added support for more custom types in SQL Server table imports.
- Cleaned up precision and scale settings on table import for types that don't support them.
- Added option to insert raw provider annotations into table import results.
- Fixed bug where importing multiple tables with the same name in different schemas wouldn't work in GetDatabaseSchema
- Added option to put imported tables into logical display folders based on database and schema

SSAS Emission

- Fixed an issue in SSAS role emission where the database write permission would be incorrectly set to Allowed instead of None.
- Fixed an issue where AnalysisServicesProcessing task was not being emitted with correct settings for Parallelism.
- Fixed an issue where database objects with special characters would not be correctly encoded in the DataSourceView during cube emission.
- Fixed issue where SSAS cube permissions were not being emitted in some cases.
- Fixed an issue where Calculations were not being emitted into SSAS cube perspectives.

SSIS Emission

- Added support for SSIS annotations to packages, containers, event handlers, and data flows for SSIS 2012 and higher.
- Added support for dataflow and controlflow path annotations for SSIS 2012 and higher.
- Added error for SSIS 2016 and higher to not use the new virtual column names for errors: 'ErrorCode - Description', and 'ErrorColumn - Description'.
- Fixed an issue where BalancedDataDistributor output paths were not being placed into an exclusion group.
- Fixed issue where __$seqval column was being added incorrectly for CDC component outputs with CDC processing mode of Net.
- Fixed an issue where the Aggregate component would use the wrong data type when averaging certain column types.
- Fixed issue where package parameters with datetime type were not being emitted with the correct format into the project deployment manifest.
- Fixed issue where binary parameters were not being properly mapped in SQL queries (e.g. ExecuteSQL with parameters).
- Added support for the LEFT string operator to SSIS expression parsing.
- Added support for package configurations even when using project deployment model. This is not a recommended solution, but it is supported.
- Fixed an issue with permissions violations when deleting temp directories used in script project compilation in some corner cases.
- Added CachedComparisonFlags to input columns in SSIS emitter language definition.
- Fixed sorting of DTSX XML elements for 2012+.
- Fixed date mapping in ADO.NET Source.
- Fixed date and string mappings in OLEDB Source.
- Fixed date mapping in ADO.NET Destination.
- Fixed date and string mappings in OLEDB Destination.
- Fixed DontSaveSensitive support in OLEDB connection.
- Stopped emitting __$seqval as an external column in CDC Source when using an incompatible CDC Processing Mode.
- Fixed issue where format file references were being improperly emitted for BulkInsert tasks.
- Fixed issues with XmlSource where outputs were not being constructed correctly for unbounded elements.
- Fixed issue with CDC Control Task where emission would fail if StateConnection or StateName were not supplied.
- Fixes for SSIS Lookup type mappings for date and text.

- Fixed an issue where errors would be produced if you used a parameterized query in an SSIS component where the variable used to set the parameter contained an expression.
- Fixed and NRE when parameters do not exist in an OLEDB source or lookup build.
- Updated emission properties to correctly handle SSIS 2008/2005 packages.
- Added XML escaping to SSIS emitter to fix 2012+ emission errors.
- Updates to SSISEmitter for 2016 support.

Javascript Emission

- Added GetJson method with several variants to every Biml object and collection.

Command Line Tools

- Changed the output path of the bimlproj that is autogenerated from the ProjectView so that it is no longer in temp and has a reasonable name.

Biml Validator

- Fixed issue where validator thread would be interrupted if a table had multiple keys with the same name but no assigned schema.
- Added a validator for AnalysisServicesProcessing task to ensure that the correct mix of properties is specified for each type of processing job.
- Updated SsisEmitter dataflow type mismatch error to actually list the mismatched types.
- Fixed an issue where user generated errors/warnings/notifications would not show up on initial project load in BimlStudio.

Internal

- All assemblies are now strong-named.
- Changed MSBuild configuration files (bimlproj) to use MSBuild version 4.0 by default.

Documentation

- Added support for building documentation to command line compiler and MSBuild.
- Added checkbox to ribbon for documentation build.
- Added documentation preview.
- Added new item templates for documentation files.
- Added CSS, HTML, and Javascript editor support to enable Documentation styles and templates.
- Added new command line and MSBuild options for tabular and documentation support.

Provider Support

- Added Db2 OLDEB provider.
- Added OpenEdge Odbc provider.
- Adding basic Excel ACE OLEDB support to GetDatabaseSchema.

Stability Improvements

- Fixed exception in BimlStudio that would occur sometimes with mouse-based copy/paste.
- Fixed exception crashing BimlStudio when writing settings files.
- Fixed an invalid cast exception that crashed BimlStudio, discovered through user crash logs.
- Fixed a crash issue due to UI thread mismatch in BimlStudio, discovered through user crash logs.
- Fixed crashing issue caused by an exception in confirmation dialogs, discovered through user crash logs.
- Fixed validation reporter bug that threw an exception when reporting messages that included guids.
- Fixing BimlStudio DockSite registration-related crashes on Forward/Back navigation.
- Fixing crashes when expanding templates into projects, discovered through user crash logs.
- Fixed issue with icon references that prevented complete uninstall.

- Changed some Linq collection Cast operations to OfType to avoid BimlStudio crashes.
- Updated RestProxy to work with transparent proxies that use LDAP credentials. This significantly improves the reliability of license checks and online content checks.
- Fixed issue with crashing metadata model property data grid due to missing CustomLength property on the property view model.
- Only referenced components in Offline Schema Items actually changed to make Component not required. This fixes some issues with transformers that cause the offline schema items to break.

GetBiml Updates

- Fixed GetBiml bug affecting singleton children.
- Fixed an issue with the emission of least qualified names for Biml object references.
- Added PropertiesToOverride support to new GetBiml method for when some properties need to emit children rather than attributes in SSIS emission.
- Fixed issue with SSAS GetBiml that was causing syntax errors.
- Adding forced GetBiml emission of required attributes.
- Fixed some code generation for GetBiml to support SSIS 2008/2005.

UI Updates

- Added code to activate BS error window when build fails.
- Updated new project screen styling and icons.
- Added icons for connecting to network in status bar and enable/diasable logging.
- Updated BimlScript ribbon tab with new icons.
- Added blue dot to Live Biml files in the Project View.
- Improved Project View tool tips for Biml files.
- Added button to delete analysis metadata objects in the table editor.
- Switched to using new icons for PackageProject, CubeProject, and TabularProject.
- Renamed "Generate Documentation" to "Build Documentation".

Package Updates

- Fixed static properties for several package tasks to match the latest options and data types
- Fixing data types for some task nodes
- Changed default SQL Server target version for new projects to 2014 and SSIS project deployment model

Bug Fixes

- Fixed issue where "Delete Unscripted" did not work for Connections in the Logical View.
- Fix for case insensitivity in VB within metadata dynamic object templates.
- Fixed issue where code directives inside includes were not being added to the parent file.
- Fixed issue where language selection in global includes would not apply to dynamic object code files.
- Fixed issue where any non-Biml files in the logical view would be removed by not deleted from the harddrive.
- Fixed issue where Bundle settings editor did not have a scrollbar.
- Fixed issue where failed builds would be reported as successful when running in unicode support mode.
- Fixed issue where dataflow and control flow edges would not render on machines with some locale settings.
- Fixed partition source selector in partition editor.
- Fixed issue where items duplicated from a BimlScript object were not fully populated and could not be modified.
- Fixed issue where custom dataflow properties for custom components could have invaid types - particularly for System.Null.
- Fixed issue where error messages for code files would not show the correct filename and line number

New Features in Mist 4.0 Update 1

Mist Designers and Editors

- Connection Designer
  - Fixed an issue where update databases in connection editors would fail if a nonexistent database was already in the combobox text field
  - Several bug fixes, particularly around ADO.NET connections
- MetadataModel Designer
  - Fixed issue where metadata model editor wouldn't let you change the name of a relationship
- Project Designer
  - Fixed bug where project deployment checkbox in project editor would still be enabled with SSIS version targets that don't support project deployment
- Package Designer
  - Fixed package details editors for Merge and Merge Join where column data grids were not synching with Biml properly
- CubeProject and PackageProject Designer
  - Fixed not showing items in logical display folders
- Biml Editor
  - Fixed issue where syntax editor would automatically scroll to caret when .NET Quick Info sessions were requested
  - Added a completion list item for `<#@ template tier"..." #>` intelliprompt to inform user that any integer value can be used for tiers
  - Fixed outlining in BimlScript editor - now code nuggets and Biml elements are collapsible

Mist Dialogs

- Logical View
  - Added tooltip to Broken Live BimlScript group to describe what it is for
  - Fixed bug with showing Broken Live BimlScripts when not actually broken - specifically when the error was due to an non-.NET code issue
  - Fixed issue where duplicating a transformer in the Logical View would crash Mist
  - Fixed bug where using Biml Editor as default logical view action would cause both Biml Editor and Visual Designer to be opened
  - Removed internal-use properties from property grid in logical view
  - Added new modes for default logical view action that also enable Biml Designer to be used
  - When creating new items in the logical view, that item is now automatically opened using whatever has been configured as the default logical view action
- Import Tables
  - Fixed issue where indexes, view info, and a handful of other items were being imported even when they were not supposed to be based on ImportOptions
  - Fixed issue where Oracle tables wouldn't import if ExcludeViews was set
- Find/Replace
  - Changed Find/Replace window so that it can also search the output pane of the Template Editor
  - Added an editable combo box to the Find/Replace window that retains history for search terms used in the current project session

Other Mist UI

- Updated splash screen
- Switched output window to use a TextBox rather than a RichTextBox. Warnings and errors are no longer colorized, but we avoid the huge performance issues and Unicode rendering problems associated with the WPF RichTextBox.
- Added support for upgrading Mist settings from build to build
- Improved IsDirty tracking on *.mst file
- Product key is now hidden in account control to prevent inadvertent key disclosure during demos or videos
- Fixed issue where "Open In Windows Explorer" from the MDI document tab would not work for BimlScript Designer windows
- Fixed a performance issue with interactive transformers
- Fixed issue with infinite loop reloading of files when there is use of side-effect-causing GetBiml in live scripts

Biml Language

- Removed Table property from CdcSourceNode. It is not used.
- Added AnalysisServicesConnectionInformation to all DBConnections for the purposes of SSAS Impersonation and other features
- Added support for new BufferMode and BufferMaxSize properties for Teradata Source and Teradata Destination in SSIS 2014
- Added DQS Cleansing Component and DQS Connection
- Added support for project level encryption and the ProjectPassword property on PackageProjects

Biml API

- Added timeout support to all of the database providers and external data access methods
- Fixed issue where properties changed via API were not being reflected in GetBiml output
- Added GetDataTable support for most major DbConnection types
- Added AstRootNode convenience collections for FlatFileFormats, RawFileFormats, PackageProjects, and CubeProjects
- Fixed issue where AstRootNode convenience collections did not support name-based indexing in Live mode
- Fixed an issue where StructureEquals was producing incorrect results due to a missing negation operator in autogenerated code

Biml Compiler

- Improved compiler version information in MSBuild and bimlc.exe
- Added ability to get machine code from bimlc.exe
- Added support for SsisExternalColumnsQueryTimeout option to globally override SSIS external metadata discovery time outs
- Fixed issue with output paths being dynamically created and references being broken in XmlSource node
- Fixed issue where MSBuild response files were not being saved with Unicode support. MSBuild requires the BOM to identify Unicode, which File.WriteAllText does not use by default.
- Fixed issue where MSBuild would hang and peg the CPU if a built-in transformer framework failed
- Added package password handling to BimlDriver and BimlEngine MSBuild task
- Fixed a bug in AnalysisServicesProcessing emitter where parallel processing order would not be handled properly in some cases
- Changed compiler/msbuild option for SSIS encryption password from PackagePassword to the more inclusive SsisEncryptionPassword
- Added warning level option (/warn=0 or /p:Warn=0) so that warnings can be turned off
- Fixed nested includes so that class type members also can nest (i.e. <#+ #> blocks)

BimlScript Errors/Warnings

- Added file name to errors/warnings created from BimlScript directives
- Added warning when build-only assets cannot be located

- Improved validation reporting in Metadata model dynamic objects
- Added an error for duplicate dataflow column nodes. This usually arises with "reserved" column names "ErrorCode" and "ErrorText"
- Fixed issue where errors in nested includes were not being reported
- Added a warning and an automatic patch of import statements that use the old Hadron namespaces
- Fixed an issue where SSIS emitter was incorrectly producing a "not supported" error on some file resources
- Added support to ValidationItems to track if they were from .NET code nugget error or other error
- Fixed sorting of line and column fields in validation grid (numerical sorting instead of lexicographic sorting)

Database Providers and Type Mappings

- Fixed a type mapping bug in XmlSource
- Fixed a bug for configurations of Oracle that do not supply IsByteSemantic with column metadata
- Fixed a few minor Teradata type mapping issues
- Fixed Oracle datatype mapping in decimals
- Significantly improved DatabaseProviderSelector to more closely match Mist 3.4. Most notably, this will fix issues where ADO.NET connections were largely ignoring the provider setting
- Fixed a crash that can occur in an internal method for creating database connection strings when the provider type of the underlying connection cannot be detected.
- Fix for Oracle schema provider logic so that fallback will use OracleConnection type rather than Odbc connection
- Fixed data type mapping on import for XML column length and sysname

SSIS Emission

- Fixed issue where Ssis2014 packages were not having their nodes sorted correctly
- Added support for EncryptAllWithUserKey and EncryptAllWithPassword to SsisEmitter
- Turned off an SSIS bug workaround in AdoNetDestination emitter for 2012+ since SSIS fixed it in the 2012 release
- Fixed CDC Source emitter to use the actual specified time out instead of a hardcoded value
- Fixed an issue with AdoNetSource emitter where it would not convert a WSTR with length greater than 4000 to an NTEXT
- Fixed bug in XmlSource where outputs were not being updated correctly - orphan objects that broke reference were being created
- Fixed minor issue with emission of corrupt packages when using PDW with a 2008 target
- Fixed issue with DataType emission in downlevel SSIS targets
- Fixed bugs in Odbc Source and Destination where properties were not being emitted with the correct notification or type translator settings
- Minor fixes for PDW and Teradata destination emission
- Added support for variables of DT_DISPATCH type in SSIS importer
- Fix for an SSIS expression evaluation error related to casting integers to date times
- Added emission of synthetic connection manager parameters to project manifest via the connection manager translators
- Fixed issue where script tasks in 2014 would ask to be upgraded by BIDS
- Fixed issue where synthetic types like (Web Servbive Type) in script project settings would produce bad code behind during build
- Fixed issue where ISPac files were losing unicode characters

SSIS Import

- Fixed scrolling for large numbers of imported assets
- Applied file extension filters to file dialog boxes
- Fixed an issue where the LabeledFileSelector control would grow forever off the edge of the dialog
- Added a warnings and errors display on import
- Added a warning when placeholder connections are created for project connections imported from a DTSX file
- Fixed issue where CDC Source components were being imported as custom components
- Fixed issue where CDC Control task import was not correctly importing the StateVariable

- Prevented duplicate import of project connections referenced by packages
- Fixed issue where DTProj files without manifests were not being correctly imported
- Fixed issue where hex codes (for things like flat file delimiters) would not be properly imported
- Added option to automatically add a Biml annotation to imported SSIS object to indicate their source package
- Added option to add annotations to destination components with unmapped external columns
- Added feature where duplicate items in SSIS import are renamed, but a Biml annotation is added with the original name
- Added duplicate detection to script projects on SSIS import
- Added trimming of variable lists in script projects in importer
- Added error when trying to import DTS script tasks and components from 2005 packages
- Fixed an issue with import options default values when being used from the API
- Fixed issue where completely empty script tasks would import but not build (since SSIS fills in the missing files with defaults on completely empty scripts)

New Features in Mist 4.0

BREAKING CHANGES

- RootNode.PackageProjects has now moved entirely into RootNode.Projects, including all namespaces. This is done to better support other project types such as analysis services projects. The mitigation is to change all instances of the tag to the tag.
    - MITIGATION: If backwards compatibility with older versions of Mist is required, the PackageProjects/Projects element name can be conditionalized based on the tools version.

- Removed QuerySource and Destination. These were abstractions that wrapped OleDbSource/AdoNetSource and OleDbDestination/AdoNetDestination in a single interface that offered the intersection of their features and automatically chose the correct component type to emit based on the referenced connection type. These language elements were not particularly useful, were very narrowly used, and their removal will prevent unintentional use by new customers.
    - MITIGATION: Change any instances of QuerySource or Destination to the appropriate OleDbSource/AdoNetSource and OleDbDestination/AdoNetDestination

- All instances of the word Hadron have been removed from the project. Practical implications include:
    - Hadron.exe is now bimlc.exe
    - Namespaces that included "Hadron" have 1:1 replacements with "Biml"
    - MITIGATION: Empty classes were added in the removed namespaces to prevent code with legacy namespace imports from breaking scripts

- AstDimensionNode and AstFactNode have been removed. SSAS metadata is now added underneath the AnalysisMetadata element of table objects
- AnalysisServices connection property OverriddenConnectionString is renamed to ConnectionString. The Provider property is removed as it is no longer needed

Build Configurations

- Transformer frameworks are now much easier to use. FrameworkSettings within a Mist project can now be selected from a dropdown in the Build Configuration (bimlproj) editor
- Auto-generated and default build configuration files now use $(MSBuildPath) instead of hardcoded absolute paths, which makes the files more portable among multiple developers
- Changed target version defaults to 2012 (from 2008) in command line compiler and MSBuild task

General UI

- Changed Mist theme to have a more modern look and feel
- True MDI. In previous versions of Mist, only a single asset could be opened for editing at a time. In Mist 4.0, as many editor windows can be opened as system memory permits.
- All dialog boxes have been updated to use modern styling
- Save project dialog includes a list of all changed files
- Multiselect delete from logical view will now show just one confirmation dialog
- Added New Item Dialog for creating items in Logical View library or Project View
- The ability to zoom in/out has been added to every editor and tool window in Mist. Zoom levels can be set via the zoom slider in the status bar or by holding the control key and scrolling the mouse wheel. The zoom level can be reset by pressing control + 0 or by pressing control and clicking the mouse wheel
- Updated Getting started items on start page
- Small fix to Mist status bar to hide line and column info when not relevant
- Added notifications window as a central repository of all Mist notifications
- Added code to more aggressively terminate background threads for faster Mist process termination
- Fixed Biml utility file expansion so that it will also work if there is no BimlScript nuggets in a file
- Improved JumpList handling so that it matches what is shown in Mist recent and pinned items lists

Ribbon

- Ribbon layout has been improved - especially in Home ribbon
- Refresh and Refresh All buttons for BimlScripts have been added to the Build ribbon, in addition to a Refresh option on context menus for Biml files and logical view assets. This is particularly useful when external metadata has been changed and BimlScripts need to be re-executed
- A toggle button has been added to the Build ribbon to enable/disable the background processing of live BimlScripts. This is useful in cases where a large number of changes need to be made to scripts that have long execution times
- Added file addition to ribbon
- Added UI and persistence support for Framework file selection in Project View and Config files. Still need to add MSBuild pieces

Logical View

- Added LogicalDisplayFolder property to all Biml root objects and library files
- Modified logical view to have groupings by Relational, SSIS, SSAS, Metadata, Library, Broken Live BimlScripts
- Added ability to convert to files from live Logical ViewReference biml script from Live
- Added ability to view common database file types in the Logical View library
- Added support for icon change when an object is stale from broken live BimlScript. This still isn't working for subclasses of child items of root nodes
- Fixed issue where broken live BimlScript wouldn't show up in cases of schema binding overrides of root collections
- Added hierarchical filtering to logical view
- Added support for saving LogicalDisplayFolder that is set on file view models
- Added feature for default logical view open behavior override

Table Import

- Table import is now much faster - up to 500X in some cases
- Table import UI is simplified

Other File Types

- Added C#, VB, SQL, MDX, DAX, and Text Editors for the new file type support

SSIS Emission

- SSIS 2014 support
- New Tasks and Components
    - CDC Control Task
    - CDC Source
    - CDC Splitter
    - PDW Destination
    - Teradata Source
    - Teradata Destination
- Added SSIS Path Annotations to dataflow and control flow paths
- Added support to SSIS emitter for emitting external columns without the corresponding output columns. This is primarily used for database datatypes that are unsupported by SSIS.
- Changed DbType to DataType in column specifications
- Added emission of ISPAC files when targeting SSIS 2012 or higher and using project deployment model
- Fixed issue where a variable defined in an event handler could not be referenced from a descendent of that event handler.
- Fixed build crash when cache connections are used on lookups
- Improved emission of connection managers so that the individual connection properties are supplied, rather than just the

connection string property
- Fixed code generation issue reported to support with the Fuzzy Lookup component
- Added support for the new AutoPassThroughInputColumns property on Fuzzy Lookup to give it a bit more flexibility
- Added ServerExecutionID to pre-defined variables list for packages in Mist
- Added support for expressions on project connections
- Added PackagePassword property to AstPackageNode for encryption support

SSAS Emission

- Added support for Analysis Services projects (dwproj files). Previously, these files were auto-generated with no support for customizations. That option is still available, but now the projects can be fully customized
- Added new OutputCubeLabel property for cube aliasing
- Fixed issue where Delete option on perspective context menu was always disabled
- Added ability to specify an MdxFile by file path in the list of calculation objects
- Changed the Target property of a cube action to be an optional property
- Added an ID property to cube dimensions so that it can be explicitly set rather than always using the Name property
- Fixed issue where dimension attribute value column mappings were not being emitted

Project View

- Better handling of files that are referenced from project file but don't exist on disk
- Prevented renaming of files to include invalid filename characters
- Prevented background change notifications from occurring when a file is not part of the project but is visible via the "Show All Files" Project View option.

Designers

- ScriptProject
  - Fixed the expanded rendering of script projects within the logical view
  - Fixed issue where BimlScript generated script projects would have a growing list of blank files in the autogenerated collection

- Table
  - Added analysis metadata management to the core table designer with support for multiple analysis metadata objects on the same table.New approach to analysis metadata

- Package Editor
  - Fixed a crashing issue when trying to drag from a dataflow edge to a node
  - Fixed issue where clicking on package design surface would not de-select the selected node (which prevented switching focus back to the node)
  - Added a link to quickly open the Package Toolbox when the package design surface is empty

- Designers synchronizing with Biml code
  - Empty parent tags are now converted to self-closing when emptying a child collection or nulling a singleton child
  - Fixed XML emission of singleton children that have existing elements (usually because the object has properties set before it is added to the parent collection)

Licensing

- Fix for issue where Mist would crash when entering an offline product key if a product key had never been previously entered.
- Fixed a spelling error on the license key page.
- Fixed issue where Mist can sometime crash when it has a license key that cannot be deserialized due to invalid UserData content

- Added ability to get machine code in account control
- Added retry button to product key entry dialog
- Converted all product key checking to be asynchronous

SSIS Importer

- Added a new SSIS Importer dialog box that enables finer grained control of the import process
- Added ability to perform an SSIS import programmatically using .NET APIs from BimlScript (similar to table import capabilities)
- Added support for importing DTProj files, ISPAC files, and project deployed to a package catalog
- Added version auto-detector to package importer
- Improvements to column mappings and reducing unnecessary property specifications.
- Fixed the importer for Fuzzy Lookup, which was basically broken for passthrough columns
- Fixed an issue where UnitSeparator (\x1F) would be incorrectly imported when used within flat file formats.
- Fixed an issue where custom log providers were not being imported

Biml Language

- Directives
    - Added designerbimlpath to the template directive so that intelliprompt can be provided correctly on Biml fragments (that are later included or called)
    - Added code directive to reference C# or VB files with code directly from Biml files. This enables a variety of scenarios around creating more complex and reusable .NET code for use within BimlScripts
    - Nested include files now work

- Other
    - Added ObjectTag property to all Biml API classes. This is essentially a .NET Dictionaryt object that enables storage of .NET objects in a similar way to annotations in Biml.
    - Added support for external script project files in the language
    - Added GetPropertyValue method to all Biml API objects. This provides another option to get object property values when doing dynamic code generation.
    - Added Parse methods to all Biml API objects. This enables easy parsing of Biml code snippets to create Biml API objects. It is the inverse of the GetBiml functionality.
    - ConstraintMode has been set to default to Parallel, so that it no longer needs to be specified for most package objects
    - Added BuildSettings object to AstRootNode to access build settings in a central location
    - Added FriendlyName convenience property to columns
    - Fixed Biml reference resolution code so that references are updated even when DisableAllChangedEventInfrastructure is set to true. This is essential to enable the use of utility methods like IsUsedInPrimaryKey within live scripts on tables also imported within the live scripts
    - Added an IsDatabaseCharSetUnicode override property to Oracle connections

Biml Code Editor

- Improved intelliprompt completion lists, especially around LINQ expressions
- Added Intelliprompt completion list the GetTag method that shows valid tag values for the target object type
- Added toggle buttons to filter intelliprompt completion lists
- Intelliprompt completion lists for file system navigation in include, code, and other directive file references
- Intelliprompt completion lists for namespaces in import directive
- Added hyperlinks in code files to quickly navigate to referenced include, code, and CallBimlScript files using control + click
- Fixed the issue where the quick info for the xmlns attribute of the Biml element would show as invalid
- Added intelliprompt for files referenced by code directives
- Added Find All shortcut binding to Ctrl-Shift-F
- Added pre-population of find terms with selected text and auto selection of find terms

- Fixed issue where switching to a different application and then back to Mist would cause code editor to lose focus

BimlScript Errors/Warnings

- Fixed an issue with ValidationReporter where it would occasionally report line numbers into our temp files. This usually happened with mismatched end braces
- Improved validator for SSIS data flow columns to show the duplicate column name when the same column is mapped multiple times
- Fixed an issue where transformer generated errors were not being reported correctly. They previously appeared as Null Reference Exceptions because of an issue in the reporting mechanism.
- Fixed issue where wrong column name was being shown in Script Component column type mismatch errors
- Clarified error message when "Build and Run" or "Build and Execute" was performed without the correct version of SSIS installed

Documentation

- Fixed spacing in table column documentation
- Fixes to documentation generator to handle illegal filesystem characters in asset names and to report the exception message on doc gen failure

Metadata

- Added capability to create metadata models
- Added auto-generated metadata UI creation based on supplied metadata models
- Added dynamic object generation for easy and portable access to metadata through BimlScript code nuggets
- Added live metadata validation errors based on validators specified in the metadata models

Offline Schemas

- Added ability to specify external metadata for use in building SSIS packages via offline schemas
- Added ability to use offline schema information in SSIS build process
- Added background tracking of offline schema information to ensure that it remains synchronized with both the packages in the Mist project and the external data sources

New Features in Mist 3.4

ist 3.4 is another successful short release cycle that delivers powerful new features - less than 6 weeks after Mist 3.3.

Mist UI Changes

- Added a forward/back button in the quick access toolbar. Forward and back can also be accessed with Alt-Left/Alt-Right, mouse buttons, or touch gestures. History is maintained for 15 items by default.
- Added a search box to Project View which enables search/filtering by file name
- Fix for missing root folders on project load
- Added context menu commands for new sql generation extension methods (see below)
- Copy Biml context menu item now on all objects. This will give you the raw Biml for any object, even if it is live scripted.
- Changed the default on logical view to remove empty folders/groups when a search is active
- Fixed a Mist crash when broken transformers are run from context menu.
- Improvements to licensing and activation - especially in offline scenarios.
- Added "View in BimlScript designer" to all logical view root items, which enables you to more easily see expanded Biml for live scripted objects

Package Editor

- Entirely new Compact Mode that can be toggled with a new ribbon button and which is saved as a project setting.
- ~20% performance improvement in Package Editor load time for big packages.

BimlEditor Changes

- Added highlighting of matching delimiters (XML tags, code nuggets, multiline comments, braces, parentheses, etc.)
- Added highlighting of other instances of the currently selected text
- Fixed intelliprompt to work better with code in included files
- Added indentation guides, which are on by default.

Language and Compiler Changes

- Added SurrogateKey as an option for table column ScdType.
- File group modeling is now available on databases. Tables and indexes can be assigned to file groups explicitly
- FilterPredicate has been added to AstIndexNode to support filtered indexes
- Added DefaultConstraintName to AstTableColumnBaseNode to support naming the default constraint. This is especially useful for source controlled DBs
- Added new properties for Package emission (VersionMajor, VersionMinor, VersionBuild, CreatorName, CreatorComputerName, CreatedDate).
- Added support for DataflowOverrides, which provide functionality for changing much of the default options that the compiler sets on output paths and columns.
- Fix to enable Text and NText in FlatFileFormats
- Fixed custom component emitter bug reported in forums.
- Fixed bug with custom log provider emission

Extension Methods

- Added extension methods to string for XmlEscape and XmlUnescape
- Added GetColumnCompareList and GetColumnAssignmentList methods on table for easier construction of complex sql queries
- New extension methods to get CreateAndDropDdl for relational object types
- New extension methods to get SELECT, INSERT, UPDATE, and DELETE query templates for tables

Package Importer

- Importer enhanced to use DataflowOverrides
- Added support to importer to disable import of SSIS descriptions as annotations. These are now on by default.
- Lookup parameterized queries now import only if cache mode != full

New Features in Mist 3.3

- ODBC
    - ODBC connections are now supported, including modeling the connections and using them to import tables
    - SSIS 2012 ODBC Source and ODBC Destination are now fully supported with native language tags (no more custom component tags)
- Attunity Oracle support
    - Oracle connections are now supported, including modeling the connections and using them to import tables
    - The Attunity Oracle Source and Oracle Destination components are now fully supported with native language tags (no more custom component tags)
    - Just like the Attunity components in BIDS/SSDT, Mist will dynamically check for the presence of the Oracle Data Access Components, and build only if they are present.
- BalancedDataDistributor
    - This is a component that is shipped by Microsoft as a separate download. It is now fully supported with native language tags
- MultiFile and MultiFlatFile Connections
    - The HasMultipleFiles boolean property has been added to the existing FileConnection and FlatFileConnection nodes. When present, the FilePath property will automatically accept wildcard syntax
- Extension Methods
    - An overload of the AstNode.GetTag method was added to support case insensitive searching for tag annotations.
    - The GetBiml() extension method has been improved for use with AllMergedColumns in CloneTables.
    - The following Column and Table properties have been enhanced to provide more accurate results: Column.IsUsedInPrimaryKey, Table.HasIdentity, Table.PreferredKey.
- Miscellaneous
    - A new comment delimiter has been added to the BimlScript language. Use <#* ... *#>. This will prevent the enclosed code from being processed in any way.
    - Execute Package task can now specify Parameter Bindings
    - Script project assembly references can now include environment variables

New UI Functionality

- Backstage Splash
    - When Mist first opens, it presents a custom screen that is optimized for new and returning users. Either select from an extensive list of recent files, or choose a new project template. The more traditional Backstage menu options are just a click away.
    - Backstage
        - New
            - Beautifully rendered and extensible project template interface
            - Project Templates are now updated via RSS feed, which can be managed by Varigence or by the customer
            - Project Templates are now zip files instead of copied folders. This enables easier deployment.
        - Open

        - Provides a variety of options for finding projects and files, including extensive history tracking
        - History is easily manageable and common files can be pinned for quick selection

    - Save
        - Files with unsaved changes are listed with options to save, discard changes, or review changes
        - Mist will work with your favorite change differencing tool to see the changes in unsaved files

    - Help
        - Provides a variety of options to get more information about Mist for solving your problems including: Get Started, Tutorials, Community Forums, and Latest News

    - Account
        - One location to get all of the information about your Mist version, plugins, product key, EULA, and more.

    - Customizability
        - All of our dynamically updating content is now based off of RSS, with a unified, multi-tier caching RSS Feed Manager. Advanced users to override the feeds to support custom content for corporate deployments.

Improved UI Functionality

- General
    - Updated almost all Mist icons to be sharper and more Modern-UI friendly.
    - When no items are present in a tree or grid control, the message now identifies the parent by name for clarity
    - When no items are present in a tree or grid control, there is now an add button with a tooltip detailing the ways to add new items
    - Mist now starts in Logical View by default. It will subsequently remember window layout of the last run.
    - Added 32-bit to the title bar when running Mist in 32-bit mode. This parallels the 64-bit label.
    - Made it so that Execute Transformers context menu is hidden on script-generated assets.
    - Visibility of the grid details pane is now collapsed when multiple items in a datagrid are selected.
    - DataGrid row hightlight style is grey instead of blue when the control is not active. This should clarify which controls have focus.
    - Dotted line around the perimeter of the control when you delete the last item is now removed.
- Tree Views
    - When you single-click an already focused tree item, it enters edit mode.
    - Expanding/collapsing a node will now ensure move it to the appropriate place within the scroll window.
    - Focus now remains consistent on expand/collapse.
    - Deleting an item now appropriately focuses the next item.
    - All trees that should be multi-select are now multi-select
- Designers
    - Package Designer
    - Breadcrumb zoom out no longer breaks node sizing
    - The package details area now contains a datagrid for package parameters, with appropriate context menus
    - When dropping a node into a package or container, the node will now always be on top (not placed underneath another node) and the obfuscation border should appear when appropriate
    - Resizing a nested node now triggers scrollbars in the package editor
    - Visual flickering when trying to move large containers and tasks is eliminated
    - Disabled nodes and containers now display a gray, partially opaque overlay.
    - Fixed minor margin and spacing issues with package details controls
    - Fixed issue where sometimes tasks would not appear on design surface until the breadcrumb root is pressed

- Made resize handle for t-sql editor more easily visible
- Buttons in the task dashboard in the package editor are now always clickable and always offer the option to create new items.
- Fixed a Mist crash when setting the aggregate column operation in the aggregate component details editor.

- Table/Dimension/Fact Designers

  - Fixed broken drag/drop scenarios with snowflake columns from dimension columns to attributes.
  - Added multiselect support to all drag/drop operations
  - Whitespace padding has been added to the bottom of the Table Editor tree views to create an unambiguous drop zone for drag/drop operations
  - The focus is now set to the object created when a column is dropped. This reduces confusion, especially when that item needs to be immediately deleted.

- Script Project Designer

  - Improved the naming of some script project context menu items.
  - Fixed the Add Connection Reference context menu in the Script Component Project Designer to always fire the necessary commands
  - Eliminated an intermittent Mist crashing issue with the script project worker thread. This could only happen when Mist was already in the final stages of shutdown, but it was a bad experience.

- BimlCode Editor

  - Made it so that we don't list attributes in Biml intellisense that have already been used.
  - Improved autocomplete behavior around delimiters.
  - Added code highlighting for directive and .NET code blocks.
  - The BimlEditor now displays line numbers in the left margin. There's a user setting in the project designer to toggle the line numbers.
  - QuickInfo popup now is closed if you switch focus to a different application
  - A syntax editor's zoom percentage is now displayed in the right side of the status bar, past the line and column information.
  - Fixed a corner case that can produce a crash in QuickInfo tooltip generation.
  - Fixed an intermittent crash when pasting code in the BimlScript editor when clipboard has object data.

- BimlScript/Template Designer

  - Add the ability to cancel the preview.
  - Added a ribbon icon to toggle auto expansion of BimlScript in the Preview Expanded Biml.
  - TargetNode dropdown in template editor is now sorted
  - Fixed an intermittent crash in the BimlScript designer background thread when closing the project.
  - Fixed an issue where intelliprompt for Directives would sometimes offer the auto-complete options for the previous directive.

- Package Project Designer

  - Add Project Parameter context menu now works like Add Variable context menu, along with inline type selection.

- Tool Windows

  - Output Window
    - Keyboard can now be used to navigate the Output window.
    - Output window now scrolls as build output is produced (provided that the window is already scrolled to the bottom).
    - Output window now is selectable and has full context menu support.

    - Expressions Window
        - Added a scrollbar to the results pane on the Expression Builder dialog.

- Project View

    - Multi-selection keyboard and mouse interaction significantly improved
    - All Project View tree item properties now have descriptions in property grid.
    - Fixed over-aggressive treeview autoscrolling behavior.
    - Build and Duplicate commands are now multiselect with detailed options in context sub-menus.
    - Mist build actions can now be set via multi-select in property grid.
    - A setting has been provided to make the source control window stay open after using the source control functionality (such as an Update).
    - Root Folder Improvements
        - Added support for labels to Root folders in project view.
        - Renaming root folders is now fully supported.
        - Root Folders are now prevented from being moved into other root folders.
        - Add Existing, Copy/Paste, Include, and Include Descendants of Biml Files will now set BuildAction to Live for standard Biml files and to Reference for BimlScripts

- Logical View

    - Logical View now fully supports multi-select
    - Added Properties context menu item for all object types in LogicalTreeView.
    - Expanded Context Menu items that are active for BimlScript generated connections
    - Script projects can now be duplicated.
    - Logical view is now sorted by ScopedName and/or LogicalName where appropriate. This is particularly useful for tables, dimensions, and facts.
    - Fixed an issue with project view and logical view scrolling so that a drag doesn't get initiated in the middle of a scroll.
    - Fixed rename button in context menu for Biml Files in library.
    - Added 'Build' and 'Build and Open' for PackageProject nodes

Build Engine

- Performance

    - Massively improved performance of escape character handling in the literal strings for SSIS expressions.
    - Massive improvement to script project compilation by batching and precompiling all of the scripts used to generate the code behind files.

- BimlScript Processing

    - Fixed issue where negative tiers were not being handled as file references in live processing engine.
    - Tier 0 references to higher tiers now are fully supported during incremental processing.
    - Fixed issue where higher tier BimlScript couldn't see lower tier assets after equal or lower tier assets are changed in live processing engine.
    - When BimlScript or Biml file is changed from Live Mist Action to Reference it is now correctly removed from the in-memory model in the live processing engine.
    - Fixed a minor issue where setting debug=true in BimlScript would always result in a compiler error.
    - Fixed a Mist crash on Project reload when the project file was updated by subversion.

- Code Generation

    - Fixed an issue with VariableValueChanged event handler emission.
    - OLE DB Source component now has the correct description in documentation.
    - Targeted changes to script project binary code emission to prevent extra binary code elements from being created

when roundtripping package through SSDT.
    - Packages with Text columns as source and destination now emit correctly (without an unnecessary NText conversion).
    - SSIS data flow path names are now scrubbed for invalid characters in task/component names
    - Attribute ordering on Package node in SSIS has been changed to avoid confusing SSDT when it tries to upgrade a Biml-generated package to SSIS2012.
    - Fixed underscore and other invalid character removal from automatically generated property names in script component code behind.
    - Added methods to correctly quote identifiers for each database SchemaProvider.
    - Updated command line documentation for hadron.exe for the buildOnly option.

- Name Management

    - Internal improvements to our NameGeneration framework
    - When editing an asset name in the property grid, the Biml file rename dialog no longer triggers unless the name actually changes value.
    - Improved file copy/paste and duplicate naming to put " - Copy" at the end.

- Validation

    - Added a warning validation for when Attribute Relationship name does not match the name of the child attribute. This is to keep BIDS/SSDT from issuing a warning.
    - Fixed bug with over-aggressive Aggregate transformation diagnostic.
    - Fixed validator to prevent it from crashing the validation thread when a package or variable name is not set.
    - Added a build-time diagnostic to validate that input buffer column types in script components match the input columns to which they are wired in the dataflow.
    - Prevented sql parse exceptions from killing the validator thread.

Import

- Package Importer

    - SSIS Importer is now smarter when importing custom properties on custom components that contain lineage IDs
    - PersistFileConnection is now imported correctly for CacheConnections
    - When Importer creates an ImportedProjectConnection, it now sets CreateInProject = true
    - Importer now handles SynchronousInputId correctly for custom components in SSIS 2012
    - OleDbSource parameters are no longer imported when AccessMode doesn't support them, even if they are cached in the dtsx.
    - Importer no longer assumes that referenced project connections are always OLE DB.
    - SSIS script project import no longer assumes that the VSTAProject name does matches the class/csproj name in the script project code. This arises when developers copy/paste code from other script component instances.
    - Package import is now limited during trial

- Table Importer

    - Primary Keys are no longer imported twice (both as a Primary Key and Index)
    - Imported views now have the CREATE VIEW statement trimmed to match view definition expectations in Biml
    - Computed column attribute is no longer added when there are no Computed values.
    - Clustered indexes are no longer imported as non-clustered indexes.
    - Fixed an issue where multiple table imports could be running concurrently by the background services engine.
    - Made schema importer more resilient to errors loading the sql parser

Installer

- Improved support for side-by-side installations with other versions and bitnesses of Mist.
- No longer installing defunct CHM file. All help is now online.

- Mist installer now advertises MIME type for mist project (mst files).

Misc

- Fix for intermittent crashing caused by Logitech gesture mouse.
- Annotation text within a package node is now trimmed to remove leading/trailing whitespace.
- Annotation data grid context will now add a new annotation rather than adding an annotation to the selected annotation.
- Fixed an issue where script component project template code was trying to be version agnostic, but BIDS/SSDT would reject the code pattern on roundtripping.
- New project dialog now focuses project name textbox by default.
- Added Ctrl+Shift+B keyboard shortcut for build.
- Reduced CPU usage of background services when Mist is idle.

New Features in 3.2

Mist 3.2 continues our commitment to frequent updates. Mist 3.2's theme is improving the user experience, and providing several enhancements. These include:

- **UI Improvements**
  - Modernized User Interface
  - *Package Designer Details*
    - Modernized Package Designer UI
    - Improved package dashboard
    - Package surface can scroll during drag and drop
    - When adding a task to the designer, vertical padding is added beneath it
  - Added visual notification to the Windows task bar when a build finishes
  - Added context menus to the Configuration designer
  - Project View multi-select works with shift+click
- **SSIS 2012**
  - Fixed emission issues with package and project parameters
  - Fixed emission issues with log events
  - Proper emission of project connections with the Execute Package task
  - Native support for Expression task
  - Added protection level to package projects
  - Added package parameters to Expression Builder
- **Package Import**
  - Import without IDs is now the default behavior
  - *The following now import correctly*
    - Log Events in SSIS 2012
    - Output paths for packages with script components
    - Packages with package parameters
    - Packages with connection names containing a backslash
    - Script tasks and components that use VB
- **Logical View**
  - Execute Transformers context menu now sorts transformers alphabetically
  - Execute Transformers context menu now displays transformers that start with an underscore
  - Fixed duplication error when duplicating a package that references a script task or component
- **Project Designer**
  - Use Project Deployment is only enabled when targeting SSIS 2012
- **Biml**
  - Added ValidationReporter.Report(IFrameworkItem, Severity, Message Recommendation) overload
  - Several improvements to error and validation reporting
- **Setup**
  - Streamlined installer
  - Eliminated the SQL Server prerequisites from Mist installations

New Features in 3.1

Mist 3.1 begins our move to a steady, and more frequent, update schedule. Mist 3.1's goals include fixing remaining usability issues and implementing targeted features:

- **Connections**
  - Connection string builder dialog added for SQL Server
- **Context Menu and UI Consistency Improvements**
  - Available BimlScript Transformers are now listed in the context menus for every object in Mist. They can be executed for that object directly from the context menu
  - In any context menu where an annotation can be added, we now let the user navigate to any annotations that are already on the item
  - Audit was performed of all context menus in Mist to improve consistency and add features
  - Context menus were added throughout the package designer and package details grids
  - *Context Menu Details*
    - All data grids now support a Copy Selection command via context menu
    - Fixed missing type icons, inability to select group items in the Partition, Aggregation, and Calculation data grids, added context menus in the cube editors and script project data grids
    - The Find Results context menu now has commands for Go to Next Location and Go to Previous Location, which matches Visual Studio
    - Additionally, the error list context menu now has an icon for the copy command
    - The References list, Output control, and Recent document context menus have been updated with commands and icons, where appropriate
    - Improvements to context menu tooltips: word wrapping and limited width, and so separators appear correctly
    - Rename command is now enabled in context menu for table column and package project parameter
    - BimlScript Designer Input Editor's context menu now has undo / redo commands
- **Project and Logical View**
  - In Project View, selecting a file and typing ctrl-c, ctrl-v now copies and pastes the file
  - In Project View, right-click and copy will add the name of the file to the text copy and paste buffer
  - In Project View, Context Menu no longer strips out the first underscore in asset's name
  - In Logical View, "Add Root Folder" is disabled in the logical context menu
- **Project Designer**
  - Textboxes now extend the width of the designer
  - The command line options textbox is now multi-line. It accepts returns and a vertical scroll-bar will appear as needed
- **Biml**
  - Biml files produced during import tables or import packages dialogs are now auto-formatted
  - Biml files and Mist project files are now saved in UTF-8 format
- **BimlScript: Language, Formatting, and Intelliprompt**
  - Added BimlScript annotation directive, including updates to intelliprompt to support it
  - Fixed autocomplete errors with BimlScript directives where an extra '=' would be added in some cases
  - Fixed some other minor Biml Intellisense issues
  - Added error squiggles to BimlScript code editor
  - Intellisense autocomplete list for tier property is now sorted numerically, not alphabetically
  - Intelliprompt no longer activates when declaring variables and within lambda expressions
  - In BimlScript Library Designer's Input Editor, the context menu's Format Biml command now performs the same operation as the Format Document ribbon button
- **Package Designer**

- Variable type items in Add Variable context menu are now sorted in ascending order, but with Empty as the first type listed
- Various consistency fixes based on a full audit of all details panes
- New ability to move Toolbox items to other groups within the Toolbox - stored in user settings
- When deleting a package, if the user pressed cancel in the confirmation dialog, we previously cleared aspects of the package UI
- The SendMail Package Details designer now display working UI for attachments

- **Script Tasks / Components**
  - Undo, ctrl-z now works in the Script Code Editor
  - The Script Source and Transformation Components now include boilerplate code for public override void CreateNewOutputRows() in Main.cs
  - ScriptComponentSource/Transformation/Destination now display ribbon buttons in package designer
  - Source Script Project: Context menus added for Output Buffers
  - Mist no longer crashes when changing the OutputBuffer Name in a biml File

- **Focus / Navigation**
  - An icon now appears to the left of a property grid identifier, indicating the control that contains the property grid's selected item.
  - The property grid title's background now becomes a light pink color if the selected item is not in a designer.
  - Added tooltip text when property grid's selected item has pink background
  - Added pink shading and icon display to the Annotations tool window

- **Infrastructure**
  - Notifications, of changes to files outside of Mist, have been improved for Windows 8 and SVN Revert
  - Exceptions produced on load of plugins are handled better and reported in detail to the user
  - SyntaxEditor context menu items are no longer disabled after a project is built (using the Build ribbon button or Build & Run)
  - Context menus now clear when build is running
  - AstTableNode.GetTableSQL() now works on a lowered AST
  - References to abstract types are no longer broken by built-in lowering transformers

- **Setup**
  - Mist can now be installed side by side with older versions

- **Miscellaneous**
  - Adding a Table View in Mist now adds the view portion to the table

New Features in 3.0

Mist 3.0 is a major update, which includes the following new features and enhancements:

- **Support for SQL Server 2012 SSIS**
- Import packages into Mist and Biml
- SSIS Project Deployment Model
- Emmit SSIS Packages
- **MSBuild Integration & Build Configurations** Mist now uses a MSBuild-based build system, enabling complete control over builds from the command line and within Mist.
- Additionally, Mist 3.0 introduces build configurations, which let you customize builds by:
    - Controlling whether built Biml files are treated as source or metadata.
    - Assigning pre and post build events.
    - Building other configurations before and after the build, thus enabling chaining.

- Build configurations can be created using Mist's new designer or by authoring them in Mist's XML editor. Learn more about build configurations here.
- **Import Tables**
- Import Foreign Key Options
    - Reference Columns with Create and Check Constraint
    - Reference Columns with Create and No Check Constraint
    - Reference Columns with Do Not Create Constraint
    - Regular Columns

- Import Views Options
    - Import views as views
    - Import views as tables

- Bug Fixes
- **Import Packages**
    - Added 2012 Package Support
    - Bug Fixes

- **Biml Language**
    - Table Columns - Added Audit Enum to SCD Type

- **All New Project View**
    - Enhanced Source Control Integration
    - Folder add, edit and delete
    - View unsaved items
    - Discard Unsaved Changes by each biml file
    - Added Mist actions for live and referenced files
    - Added Build Options previously only in logical view
    - Add root folders to projects for code and metadata reuse
    - Added the following items to the context menu:
        - Open
        - View in BimlScript Designer
        - Execute BimlScript
        - Mist Action
        - Discard Unsaved Changes
        - Build Asset
        - Build Asset & Run
        - Build Asset & Open in BIDS

        - Include Descendents in Project
        - Open in Windows Explorer
        - Delete
        - Lock
        - Remove Missing Items
        - Persist Path As

- **Ribbon**
- Added Import Tab
- Added Build & Deploy Tab
- **Improved Build Performance** Build times are 10-20% ?faster than previous releases.
- **Subscriptions** In addition to purchasing Mist, you can now subscribe to Mist at a low monthly rate. See your options at the Varigence store.
- **Other**
    - Enhanced Find and Replace
    - Side-by-Side Install
    - Bug Fixes

New Features in 2.0

Mist 2.0 and Biml 2.0 add several important new features to the available functionality.

- **BimlScript Intelliprompt** BimlScript now supports intelliprompt inside of C# code blocks. It supports all the standard C# functionality, as well as the Biml language model.
- **Import Existing Packages** Existing SSIS packages can be imported into a Mist project as Biml code. This enables developers to leverage existing investments in SSIS packages, while gaining the benefits of working with Biml.
- **Relational Model Updates** The relational model in Biml 2.0 has been enhanced to support relational partitions. It has also been upgraded to better support for multiple databases in the same project.