onestream

BI Blend Design and Reference Guide

9.2.0 Release

Table of Contents

BI Blend Overview

BI Blend is a read-only aggregate storage model designed to support reporting on large volumes of data that are not appropriate to store in a traditional OneStream cube. BI Blend data is large in volume and often transactional in nature. For example, to analyze data by invoice, a standard cube would require metadata to store the data records. In a short period of time, most of the invoice metadata would be unneeded because of the transactional nature of the data. Therefore, storage in a cube design is not a best practice solution for transactional data.

A key challenges of reporting on Transactional data, is presenting it in a uniform format that supports standardized reporting yet is flexible enough to support ever changing records and reporting requirements. The overall large size of the data sets requires a model suitable for responsive reporting and analysis. BI Blend approaches these challenges in an innovative way by rationalizing the source data for uniform and standardized reporting, much like the Standard OneStream Cube Models, but storing the data in a new relational column store table for responsive reporting.

The BI Blend solution is intended to support analytics on large volumes of highly changing data, such as ERP system transaction data, which typically would not reside in a OneStream cube. The processing is unencumbered from the intensive audit controls within a traditional Consolidation Cube, such as managing calculation status.

Key Elements of BI Blend

- Flexible for change
- Fast Aggregation through data as Stored Relational Aggregation
- Single Reporting Currency translation

- LeveragedOneStreamMetadata, Reporting and Integration tools
- Non-cube, executed to a relational table optimized for reporting on large data sets by storing results in a column store index

BI Blend Features

BI Blend is a blend of multidimensional and transactional data. This is done by utilizing the OneStream platform application to generate a database structure for OLAP reporting. By utilizing the existing OneStream Workflow processes, users need to be familiar with the interface and tools required to develop BI Blend reporting solutions.

BI Blend provides focused reporting tables that are aggregated and saved as stored parent intersections for fast reporting at a later point in time. BI Blend does not replicate an entire cube. It focuses on specific reporting use cases that result in many parent intersections that would not perform well under Calc-On-Fly aggregation.

BI Blend also solves for use cases that are not pure analytic reporting problems. By leveraging OneStream hierarchies along with BI Blend configuration settings, it is possible to aggregate on a few dimensions, for example entity or account, while including transaction information that is not associated with a cube, like an invoice number. The ability to combine the dimensional structure with transaction details enables selective enrichment of transactional data.

BI Blend is designed to support analytic models where some level of aggregation may be required, but contains constantly changing information that should not be loaded to a cube. This includes the following items:

- Read only access data
- Very large data sets that are transactional in nature
- Fast aggregation as stored relational aggregation

- Flexible and changing records
- Data related to cube data
- Data populated and rebuilt on demand
- Ancillary and supplemental data

BI Blend Stage Cache Engine

BI Blend uses in-memory processing and does not write-records to stage tables. All results are output to the designated target database. This engine is used to rationalize the transactional data by leveraging the OneStream Metadata Engine. During processing, the BI Blend Cache utilizes the cube dimensions to transform the records into a unified reporting format. The cube hierarchies are used to derive aggregation points to be stored for reporting.

The BI Blend Stage Cache also accepts properties on Entities and Accounts to perform direct method translation to the application reporting currency. The engine ultimately generates a finalized table with a Column Store Index, creating a structure for OLAP reporting. The table can be accessed with the standard OneStream BI Dashboard and Pivot Grid reporting tools for analytic reporting as well as relational blending into Cube Views and dashboards.

The BI Blend Stage Cache Engine supports:

- Hierarchy Aggregation without complex calculations
- Simple currency translation using the Direct Method only. Any destination currency can be defined, but only one per BI Blend process.
- Limited use of member properties
- Does not use dimension relationship properties such as Aggregation Weight, Percent Consolidate, or Percent Ownership
- Limited parent level calculations using derivative rules

- Basic time math using helper rules
- Supports simultaneous multi-period data loads by record for up to 36 periods using Attribute Value dimension members

The Stage Engine integrates transactional data. Source data is processed through the standard Stage Engine, where the data uses the common OneStream tools to parse and transform records. BI Blend source data contains additional members that are defined in the integration as Attribute and BI Blend Extended Dimension members. Stage Derivative Rules enhance the data with groupings available in reporting.

BI Blend Aggregation

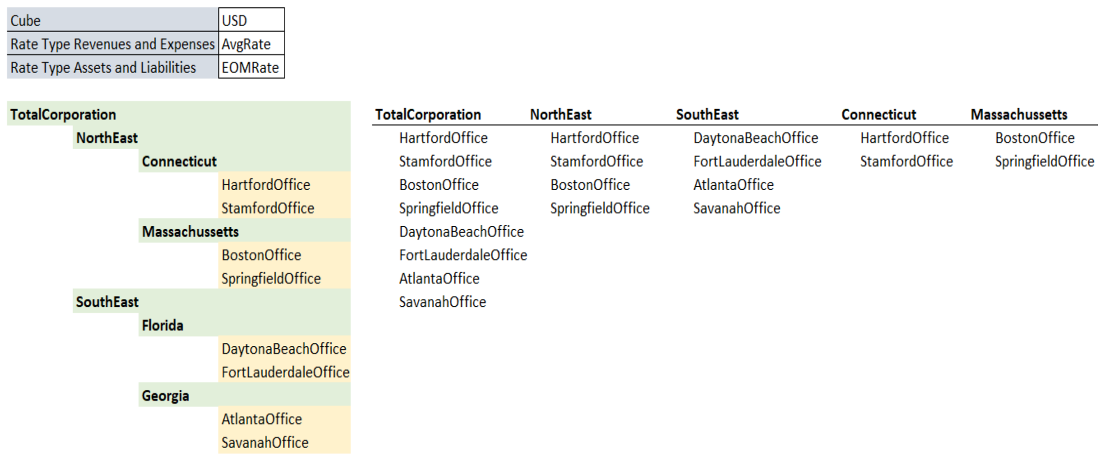
Aggregation is a simple aggregation based on the cube dimension hierarchy. Additional aggregation points for non-cube members or attribute members can be included in the results using Derivative Transformation Rules.

The BI Blend Engine aggregation utilizes the cube dimension hierarchies. This is done by evaluating parent members and identifying all Base members within the hierarchy. These base members are aggregated to the parent. Each parent within the hierarchy is evaluated using the same methodology. BI Blend processing has no concept of sub-parent roll-ups. Each parent is evaluated and aggregated according to the base members within its hierarchy.

Entity properties such as Percent Consolidate and other dimensions' Aggregation Weight are not used in the BI Blend processing. The aggregation is derived strictly from each parent as a sum of its base members. Therefore, duplicate members within a hierarchy (shared members) should be avoided to eliminate double-counting of results.

BI Blend supports changing the Blend Unit from Entity as the page dimension to any other dimension in the cube. This requires that currency translation be defined as a simple translation based on the Entity's local currency. This is the case whether Entity is set as the Blend Unit or a different dimension is used. This enables correct results when the Entity is not the Blend Unit, by providing a common currency throughout the aggregation levels to yield correct results.

Each defined aggregation is stored in a cache and each can be calculated independently.



Cube	USD
Rate Type Revenues and Expenses	AvgRate
Rate Type Assets and Liabilities	EOMRate

- **TotalCorporation**
  - **NorthEast**
    - **Connecticut**
      - HartfordOffice
      - StamfordOffice
    - **Massachussetts**
      - BostonOffice
      - SpringfieldOffice
  - **SouthEast**
    - **Florida**
      - DaytonaBeachOffice
      - FortLauderdaleOffice
    - **Georgia**
      - AtlantaOffice
      - SavanahOffice

TotalCorporation	**NorthEast**	**SouthEast**	**Connecticut**	**Massachussetts**
HartfordOffice	HartfordOffice	DaytonaBeachOffice	HartfordOffice	BostonOffice
StamfordOffice	StamfordOffice	FortLauderdaleOffice	StamfordOffice	SpringfieldOffice
BostonOffice	BostonOffice	AtlantaOffice		
SpringfieldOffice	SpringfieldOffice	SavanahOffice		
DaytonaBeachOffice				
FortLauderdaleOffice				
AtlantaOffice				
SavanahOffice				

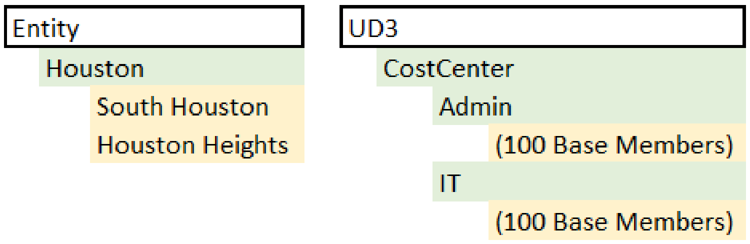
Blend Unit

Determining which dimension should be defined as the Blend Unit is a key decision which will affect the performance of the BI Blend process and the reporting results.

The Blend Unit acts to define the dimension which is used to break down the data into effective pages for processing, or partitions. Blend Unit’s page are used to process the aggregations defined in the BI Blend settings, such as Account or UD aggregations.

Each Blend Unit member's aggregations are executed as part of a multi-threading process. The selection of a dimension as the Blend Unit can impact the performance of the application. The larger the number of Blend Unit pages, the more opportunity there is for multi-threading tasks to be initiated.

Examples



Entity	UD3
Houston	CostCenter
South Houston	Admin
Houston Heights	(100 Base Members)
	IT
	(100 Base Members)

- An entity structure with three members requires all aggregation to happen within only three members.
- Assigning the larger Dimension as the Blend Unit, such as UD3 (200 members) as the Blend Unit, would allow multi-threading to process more aggregations on smaller datasets.
- Larger Blend Unit members enhance performance through a more even distribution of records.
- When a Blend Unit page completes the aggregation process, the engine loops over the rows on the page and summarizes any duplicate rows, but not with duplicates created in another page.
- Derivative rules run on each page, for a single row, in an exclusive manner for that page and do not cross pages.
- Many members in the Blend Unit means smaller pages which leads to better memory management, faster aggregation performance, and more parallel processing.

Performance Settings

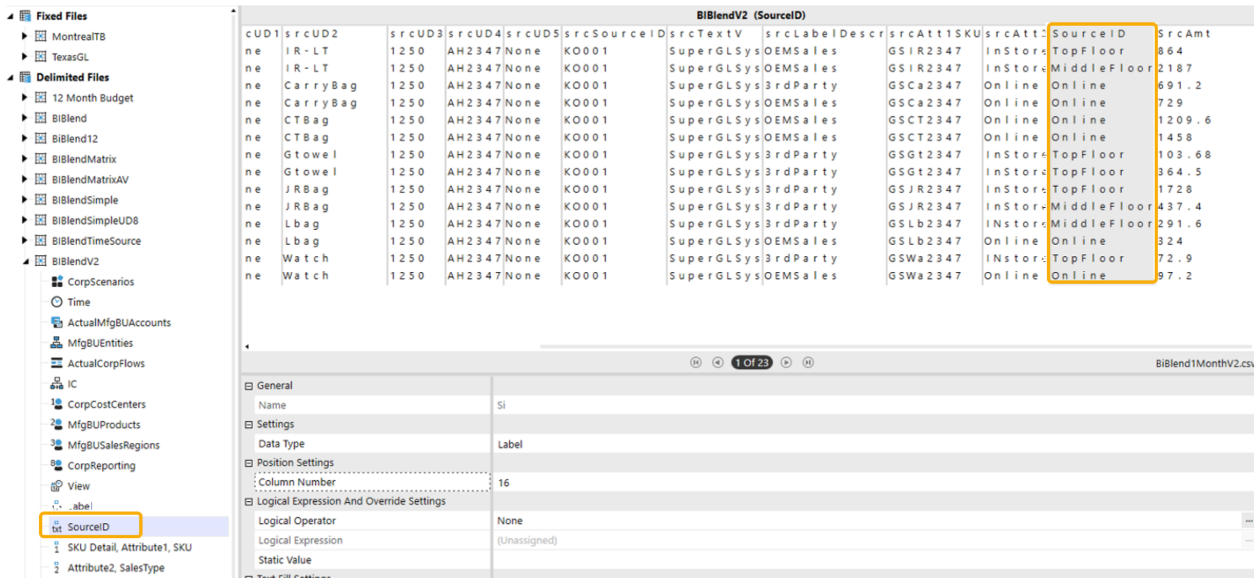
BI Blend processing is a CPU and memory intensive process. The number of table records is heavily impacted by the Attribute details in the records and level of aggregations defined for cube dimensions. In each BI Blend configuration setting, the performance can be tailored to the environment with the Performance Controls properties. See Performance Controls Settings.

Blend Unit Partitioning

The concept of partitioning a Blend Unit is a performance solution for large data sets aggregating within a Blend Unit. This permits multi-threading of the aggregations within the Blend Unit.

Blend Unit partitioning is accomplished by assigning the Data Source's Source ID property to a record field. The members within this field will be used as the key to partition aggregations within the Blend Unit page. The resulting records will be not be summarized across the Source ID/Blend Unit partition.

IMPORTANT: BI Blend Derivative Rules cannot reference data in other partitions. If calculations are required to cross partitions, then these will need to be performed in the source data set or in the reporting layer.



BIBlendV2 (SourceID)

cUD1	srcUD2	srcUD3	srcUD4	srcUD5	srcSourceID	srcTextV	srcLabelDescr	srcAtt1SKU	srcAtt	SourceID	SrcAmt
ne	IR-LT	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSIR2347	InStor	TopFloor	864
ne	IR-LT	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSIR2347	InStor	MiddleFloor	2187
ne	CarryBag	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSCa2347	Online	Online	691.2
ne	CarryBag	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSCa2347	Online	Online	729
ne	CTBag	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSCT2347	Online	Online	1209.6
ne	CTBag	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSCT2347	Online	Online	1458
ne	Gtowel	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSGt2347	InStor	TopFloor	103.68
ne	Gtowel	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSGt2347	InStor	TopFloor	364.5
ne	JRBag	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSJR2347	InStor	TopFloor	1728
ne	JRBag	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSJR2347	InStor	MiddleFloor	437.4
ne	Lbag	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSLb2347	INstor	MiddleFloor	291.6
ne	Lbag	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSLb2347	Online	Online	324
ne	Watch	1250	AH2347	None	KO001	SuperGLSys	3rdParty	GSWa2347	INstor	TopFloor	72.9
ne	Watch	1250	AH2347	None	KO001	SuperGLSys	OEMSales	GSWa2347	Online	Online	97.2

BiBlend1MonthV2.csv

⊟ General	
Name	Si
⊟ Settings	
Data Type	Label
⊟ Position Settings	
Column Number	16
⊟ Logical Expression And Override Settings	
Logical Operator	None
Logical Expression	(Unassigned)
Static Value	

BI Blend's assignment of a Blend Unit and aggregation generates pages, which are visible in the Log File. Each Blend Unit Page cannot exceed 2gb. If the 2gb size is exceeded, the aggregation will fail and the following error message will display: Error Array Dimensions Exceeded Supported Range. If a solution is impacted by size constraints, it is to increase the size of the Blend Unit dimension being used or reduce the number of dimensions used in BI Blend.

BI Blend Configuration Overview

BI Blend is designed around all the common elements used in all OneStream applications. Users familiar with setup, workflow, and reporting should be comfortable using BI Bend, as it utilizes the following functionality:

- Data Integration
- Metadata
- Data sources
- Transformation Rules
- Derivative Rules
- Workflows
- Reporting tools and dashboards

BI Blend Essential Design Considerations

This section provides information about the use of data records and common members.

Understand Data Records

- Any member not within the data record aggregation path will be bypassed. This means that a datasource can contain a complete set of records, and BI Blend can bypass those records by filtering or selecting an aggregation point that excludes those record sets.

- Attribute Members can be aggregated by being associated as a base member in a UD8 hierarchy. If the source record Attribute does not find a base target UD8, no error message is presented, and the record is ignored. This allows the data set to be flexed easily to adapt to reporting changes. Users should review the messaging for bypassed Aggregations. If the member is not within the aggregation path, it will be ignored.

Use of Common Members

Common Members reference metadata designs which have the same member names across dimensions. An example of Common Members is when Dimensions such as UD2 and UD3 both have members called Top with children as None.

- Proceed with caution when using common members (Top/None) across dimensions. Common members may cause inconsistent results.
- If fully summarized intersections are required, the designer should consider selecting another Blend Unit.
- Limiting a Blend Unit to a member may not be optimal for BI Blend processing but will yield fully aggregated dimension results.
- If duplicate records are encountered, accept the duplicates but use aggregation queries when you consume or query the BI Blend table. Group By on dimensions while performing a sum on Measures.
- Unique top members across dimensions are preferred. Change dimension aggregation information to pick a parent that does not include common members, such as None or Top.

BI Blend Processing and Performance

The requirements related to the BI Blend environment vary widely by the volume of source records together with the BI Blend settings definition.

Default Server Selection

BI Blend processes are queued across the available Stage servers. On the BI Blend workflow settings, a defined server can be assigned to dedicate all BI Blend processing.

> **NOTE:** This should only be used for Platform Versions 7.4.4. and earlier. Platform Versions 8.0.0 and later should not have defined servers.

Learning Mode

Learning Mode occurs during the first process instance of BI Blend and the design related to the choice of Blend Unit. This mode restricts multi-threading to two threads by two Blend Units to generate predictive statistics based on the number of records generated from the BI Blend settings. Subsequent processes are optimized to multi-thread each Blend Unit. Should the Blend Unit be changed, the Learning Mode is enabled again. Additionally, if the number of aggregating dimensions is increased from the prior settings, the Learning Mode is enabled again.

- Only two threads run
- Default mode when BI Blend Task is run for the first time
- Limited multi-threading is done to help ensure free memory is not exceeded

Log File Statistics

- Blend Unit
- Base Rows
- Parent Factorial
- Explosion Factor

Second Pass Processing

After a successful import in Learning Mode, the same thread can be evaluated. BI Bend processing will observe the current number of rows in each Page dimension (Blend Unit) and apply the Explosion Factor to determine if the process will exceed the amount of free memory available on the server.

If a new Blend Unit is added to the file, the calculation estimations for logging and memory usage will use an average across the Page member statistics in its calculations.

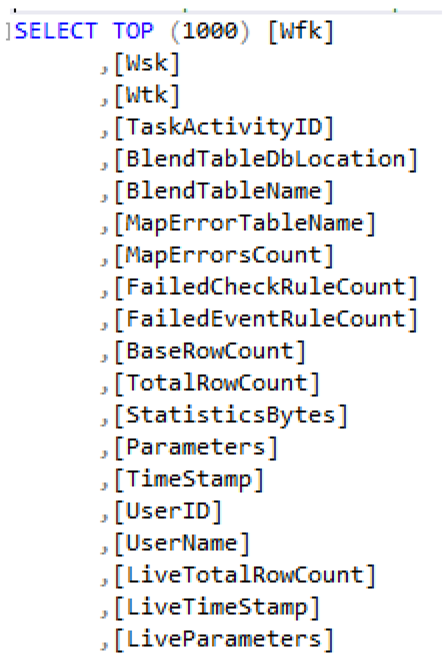
BI Blend Database Table Creation and Structure

The output of BI Blend is SQL database tables in a column store index format and views. These have a high level of compression, which optimizes them for reporting. The data is highly structured and contains the parent member values similar to cubes. The BI Blend process creates the data tables, views, and corresponding error table automatically.

OneStream Application Database Tables

A database table called the StageBiBlendInformation table captures the activity related to BI Blend tasks. This table has an associated MethodQuery in a Workspace Data Adapter called BIBlendInfo.

- The StageBIBlendInformation table is a table generated in the OneStream application database tables to manage the tables generated through the BI Blend workflow process



```
SELECT TOP (1000) [Wfk]
      ,[Wsk]
      ,[Wtk]
      ,[TaskActivityID]
      ,[BlendTableDbLocation]
      ,[BlendTableName]
      ,[MapErrorTableName]
      ,[MapErrorsCount]
      ,[FailedCheckRuleCount]
      ,[FailedEventRuleCount]
      ,[BaseRowCount]
      ,[TotalRowCount]
      ,[StatisticsBytes]
      ,[Parameters]
      ,[TimeStamp]
      ,[UserID]
      ,[UserName]
      ,[LiveTotalRowCount]
      ,[LiveTimeStamp]
      ,[LiveParameters]
```

BI Blend Database Tables

The BI Blend assigned database will have tables created for each workflow task by Workflow Tracking and Input Frequency. All tables created follow the same naming convention with different suffixes depending on the contents of the table.

Naming Convention	Description
BIB_<Application name>_<Import Workflow Profile name>_<Scenario>_<Time Period>	Tables to store the blended data.
BIB_<Application name>_<Import Workflow Profile name>_<Scenario>_<Time Period> _ Dim<Ac>	Tables to store dimension members.

Naming Convention	Description
BIB_<Application name>_<Import Workflow Profile name>_<Scenario>_<Time Period>_ Dim<Aggregation dimension token>	Tables to store the account dimension and the aggregated dimension members.
BIB_<Application name>_<Import Workflow Profile name>_<Scenario>_<Time Period>_ME	Tables to store metadata errors. Each time a table is created, a matching error table is created. The error table is created whether or not an error is present.
BIB_<Application name>_<Import Workflow Profile name>_<Scenario>_<Time Period>_RS	Tables used during the blend process to hold raw schema data. Once the import is complete, the data from this table is cleared.

Workflow and Cube Settings

The cube dimensions assigned to the cube are used by the BI Blend Stage Cache Engine to define the target for transforming records. The assigned cube dimension’s hierarchies generate the aggregation points for the resulting BI Blend relational table. This is a primary difference between the BI Blend Model and a Standard OneStream Cube Model. Within each Data Unit, the parent levels, such as an Account parent, are derived dynamically in the Standard OneStream Cube Model. The BI Blend Model derives these totals and saves them as stored values in the output relational table depending on the aggregation points defined.

The interface to the BI Blend Model is the OneStream workflow interface. Workflow is a key element of BI Blend. Workflow BI Blend settings establish the cube dimensions used to derive the metadata and aggregation points in the resulting BI Blend relational tables.

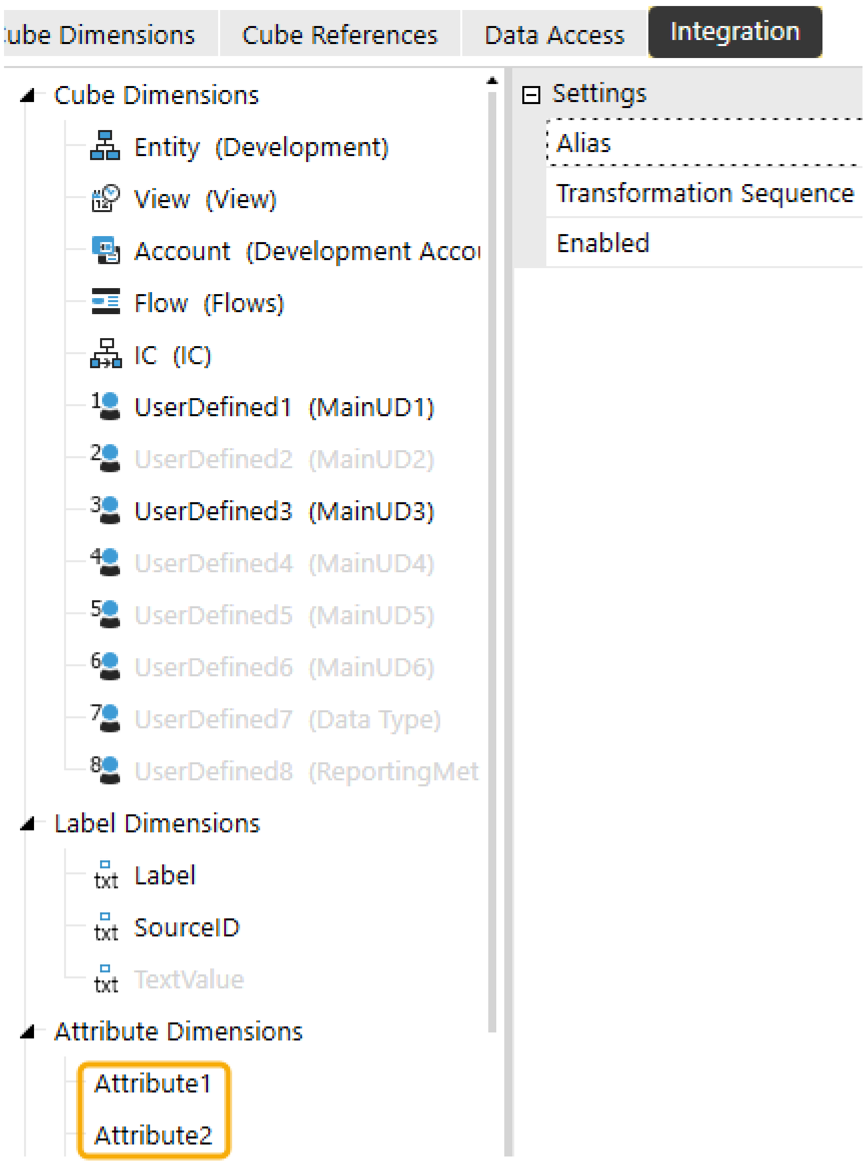
The data integration used by the BI Blend Engine uses dimensionality based on cube dimensions, stage attributes, and BI Blend extended dimensions. This array of properties, which is available for data integration, lets the designer enrich the source data for reporting.

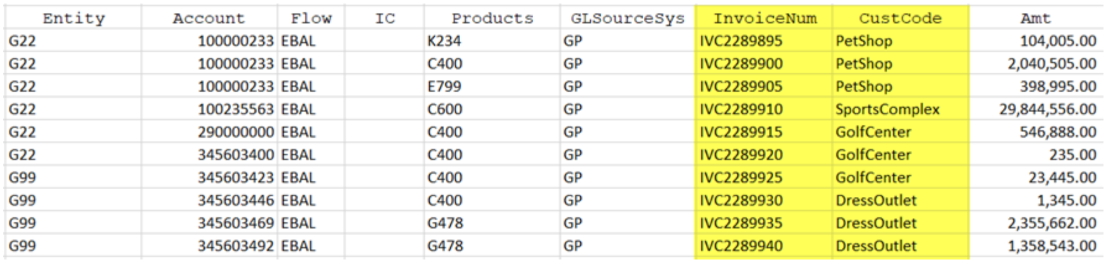
BI Blend extended dimensions are a component of the BI Blend engine Blend X extended architecture, which consists of three dimensions that provide an additional 57 fields to capture text, numeric values, and time. The Blend X extended architecture uses the Attribute DateTime dimension to deliver time-awareness to the data set, which is then used by calculations to enrich data for reporting:

- 13 cube dimensions let transformation rules align source records to cube data and to generate aggregation points based on the dimension hierarchies.
- 20 stage text attributes text records support cube dimension aggregations when the records are aligned with metadata in a User Defined 8 dimension. They do not support the use of Transformation Rules.
- 12 stage value attributes support numeric records.
- 3 label dimensions are text-based fields.
- The SourceID field is the key record attribute used to partition the records for multi-threading aggregations and for the processing of derivative rules.
- 13 Blend X Attribute DateTime Dimension fields are designed to capture time-based fields. The xBlendDateTime field usage is extended to function as the time-based property in Derivative Rule computational math functions.
- 20 Blend X text attributes expand the scope of the data set.

- 24 Blend X value attributes capture numeric data or time-series values in a matrix load format.

In this example, the Integration settings on the cube are enabled for Attribute1 and Attribute2 to expand the details to be collected for BI Blend analytic reporting, such as an Invoice Number and Customer Code.





Entity	Account	Flow	IC	Products	GLSourceSys	InvoiceNum	CustCode	Amt
G22	100000233	EBAL		K234	GP	IVC2289895	PetShop	104,005.00
G22	100000233	EBAL		C400	GP	IVC2289900	PetShop	2,040,505.00
G22	100000233	EBAL		E799	GP	IVC2289905	PetShop	398,995.00
G22	100235563	EBAL		C600	GP	IVC2289910	SportsComplex	29,844,556.00
G22	290000000	EBAL		C400	GP	IVC2289915	GolfCenter	546,888.00
G22	345603400	EBAL		C400	GP	IVC2289920	GolfCenter	235.00
G99	345603423	EBAL		C400	GP	IVC2289925	GolfCenter	23,445.00
G99	345603446	EBAL		C400	GP	IVC2289930	DressOutlet	1,345.00
G99	345603469	EBAL		G478	GP	IVC2289935	DressOutlet	2,355,662.00
G99	345603492	EBAL		G478	GP	IVC2289940	DressOutlet	1,358,543.00

Set Up BI Blend

Designing the BI Blend workflow definitions should be determined by an understanding of how the data will be consumed and analyzed in reporting. The common use of BI Blend will fall into requirements within the models of Transaction Analysis and External Data. However, it can be considered to support an aggregation model where the BI Blend aggregation processing may be a preferable solution of the standard cube design.

Define Aggregation and Report Points

The BI Blend Engine can leverage the hierarchical structure of OneStream dimension metadata hierarchies to generate aggregations. Hierarchy parent levels can be created as stored records in the resulting BI Blend tables, adding to the final number of records generated. BI Blend records processed will use transformation rules targeting a cube's assigned dimension.

The dimensions assigned to the integration used by BI Blend vary by Scenario Type, therefore resulting in BI Blend aggregation that can differ from that used in other reporting cubes. A dimension and its hierarchy can be unique for BI Blend reporting.

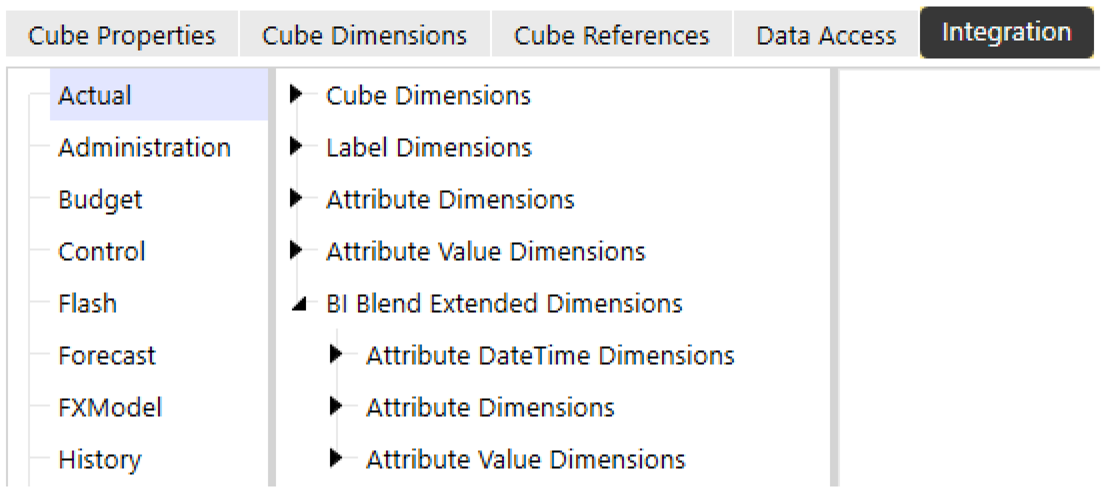
BI Blend requires a cube to determine assigned dimensions. Dimensions by Scenario Type or a specialty cube functioning as a dimension outline can be used to yield alternate reporting results that may differ from standard application cubes.

The BI Blend Stage Cache Engine functionality supports:

- Hierarchy Aggregation, no complex calculations
- Simple currency translation using the Direct Method only. Any destination currency can be defined, but only one per BI Blend process.

- Limited use of member properties
- Does not utilize dimension relationship properties, such as Aggregation Weight, Percent Consolidate, or Percent Ownership.
- Limited parent level calculations using Derivative Rules
- BI Blend Extended Time Math using Derivative Rules
- Supports simultaneous multi-period data loads by record for up to 36 periods using Attribute and BI Blend Extended Dimensions.

BI Blend Dimension Property Usage



Dimension	BI Blend Property Usage
Scenario	• The scenario is defined in the Dimension Library and a Scenario Type is selected. • Workflow Tracking Frequency • Input Frequency
Time	Time defined in the BI Blend table name is driven by the scenario's workflow tracking and input frequency or by how time is defined in the source data set. Time can be mapped as a matrix load using the attribute value attributes and the Blend X value attributes.
Entity	• Currency is referenced for simple translation • No other member or relationship properties are used • Aggregation weights are not used • Alternate hierarchies will be double counted • BI Blend does not utilize Entity Relationship Properties.Therefore, Entities as a shared member is not supported.
Account	Account Types is used for hierarchical aggregation and translation rates

Dimension	BI Blend Property Usage
Flow	• Aggregation weights are not used • Alternate hierarchies will be double counted • Does not recognize Flow Members impact of Switch Type on Account Types • Does not recognize Flow Members impact of Switch Sign • Does not support complex currency calculations or alternate input currencies • No other Member or Relationship Properties are used
User Defined	• Aggregation weights are not used • Alternate hierarchies will be double counted • No other Member or Relationship Properties are used
User Defined 8	The UD8 Dimension may contain members, organized in hierarchies, which can be used to define aggregation points for attribute members contained in the source data set.
Other Dimensions	
Consolidation	Is limited to functionality for Local and Translation. The results for translation will be limited to a single target reporting currency per BI Blend process. Complex translation is not supported and only the Direct Method is used for all RateAccount Types.

Dimension	BI Blend Property Usage
Origin	The Origin member is only supported for data generated through the Workflow BI Blend Engine as the Import member
View	View is always periodic for a BI Blend import. You can derive YTD data using reporting tools, time math business rules, or derivative rules. Accumulating results, for YTD reporting, must be done using derivative rules, business rules, or in the reporting tools as a calculation. The output results of BI Blend include an identifying column identifying the account type to identify flow and balance type accounts.
ICP	ICP Partner detail can be included in the data records, but Eliminations are not performed.

BI Blend and Cube Settings

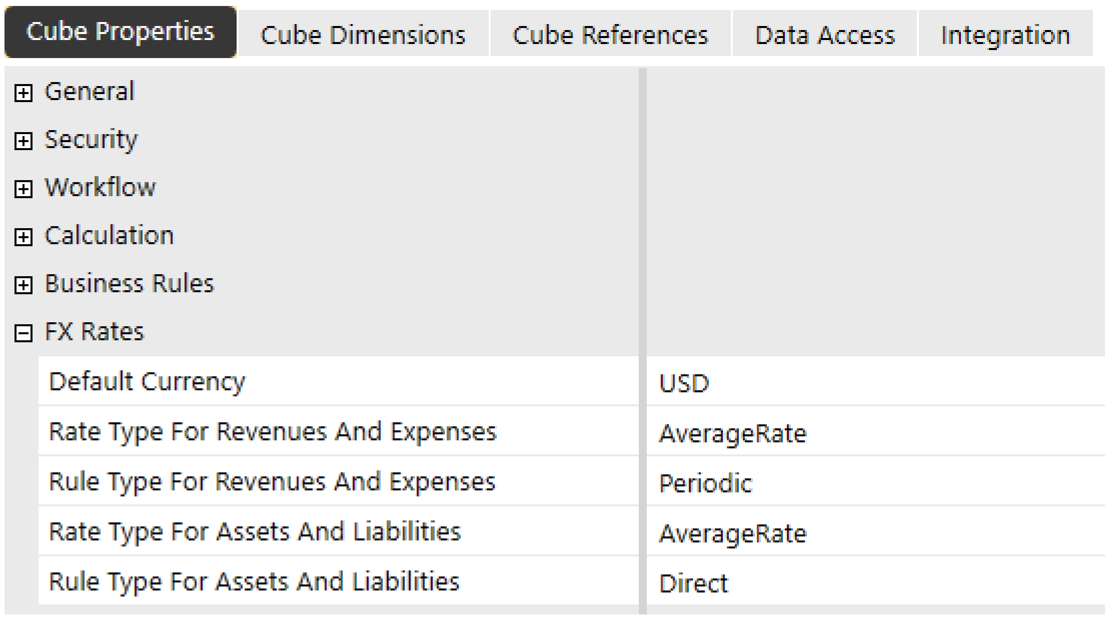
A key element of the BI Blend solution is to rationalize data for BI Reporting. The BI Blend Reporting solutions must be tied to a workflow and then assigned a cube.

Cube Properties

Currency translation is primarily controlled by the Cube Properties. Rule Type is not used. All translations use the Direct Method. The rate is determined by Account Type.

RateType for Revenues and Expenses
The rate used by Account Type property.

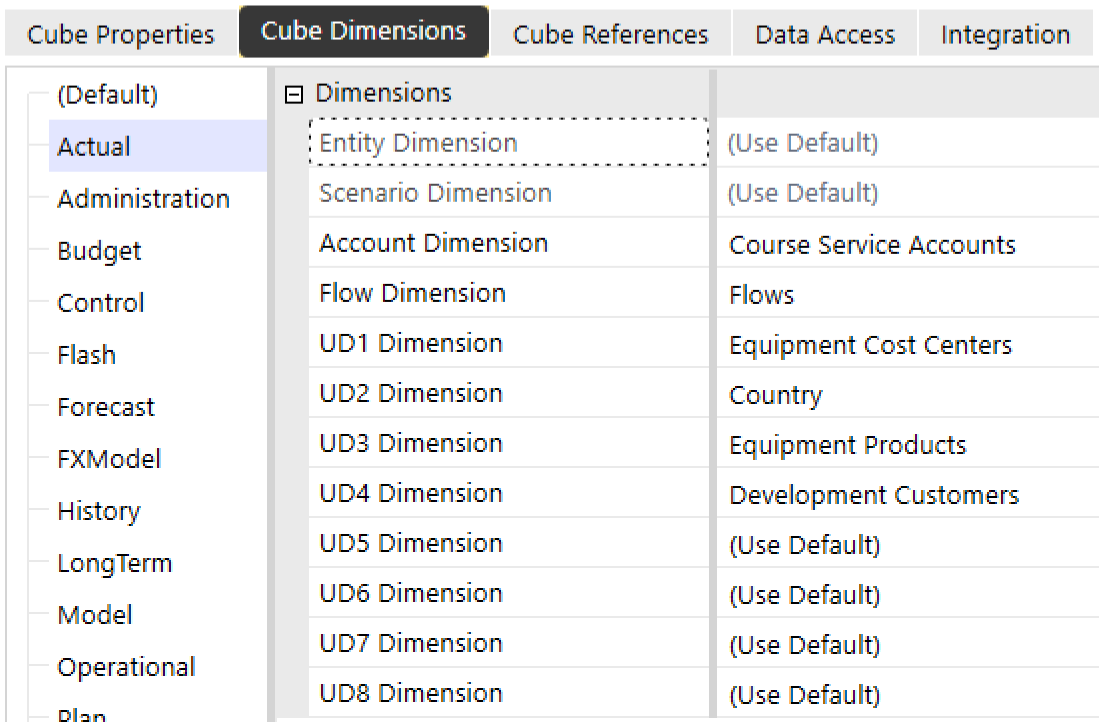
RateType for Assets and Liabilities
The rate used by Account Type Property



Default Currency	USD
Rate Type For Revenues And Expenses	AverageRate
Rule Type For Revenues And Expenses	Periodic
Rate Type For Assets And Liabilities	AverageRate
Rule Type For Assets And Liabilities	Direct

Cube Dimensions

The cube dimensions are key to BI Blend results. The assigned dimensions, by Scenario Type, are used to rationalize the data for reporting and derive the reporting subtotals.

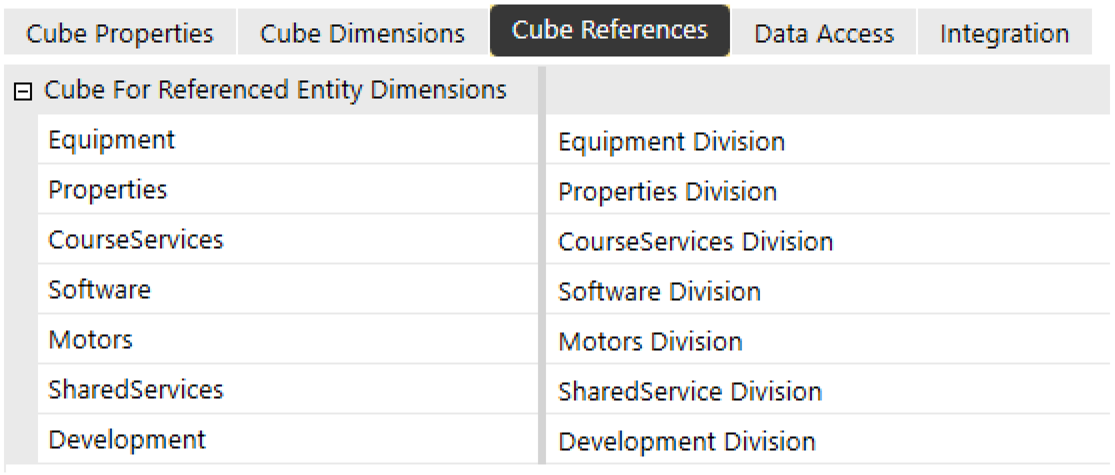


Dimensions	
Entity Dimension	(Use Default)
Scenario Dimension	(Use Default)
Account Dimension	Course Service Accounts
Flow Dimension	Flows
UD1 Dimension	Equipment Cost Centers
UD2 Dimension	Country
UD3 Dimension	Equipment Products
UD4 Dimension	Development Customers
UD5 Dimension	(Use Default)
UD6 Dimension	(Use Default)
UD7 Dimension	(Use Default)
UD8 Dimension	(Use Default)

Cube References

Cube references, and the associated Extensibility they manage, are fully supported by the BI Blend Engine. Extensible Dimension hierarchies are used if BI Blend is based on a top-level cube.

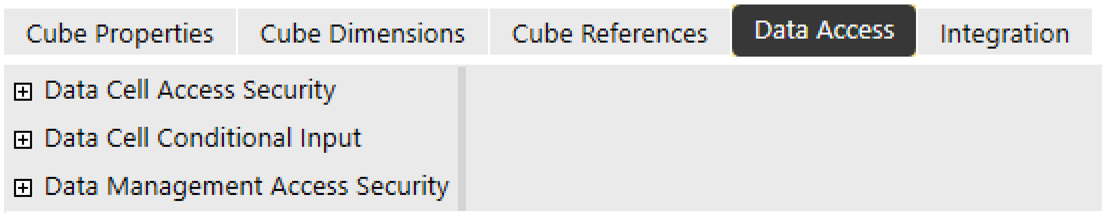
Use of a top-level cube to load to extended cubes and dimension only supports two levels of Extensibility, the first being the cube's main dimension and the second being the next level extended dimension.



Cube For Referenced Entity Dimensions	
Equipment	Equipment Division
Properties	Properties Division
CourseServices	CourseServices Division
Software	Software Division
Motors	Motors Division
SharedServices	SharedService Division
Development	Development Division

Data Access

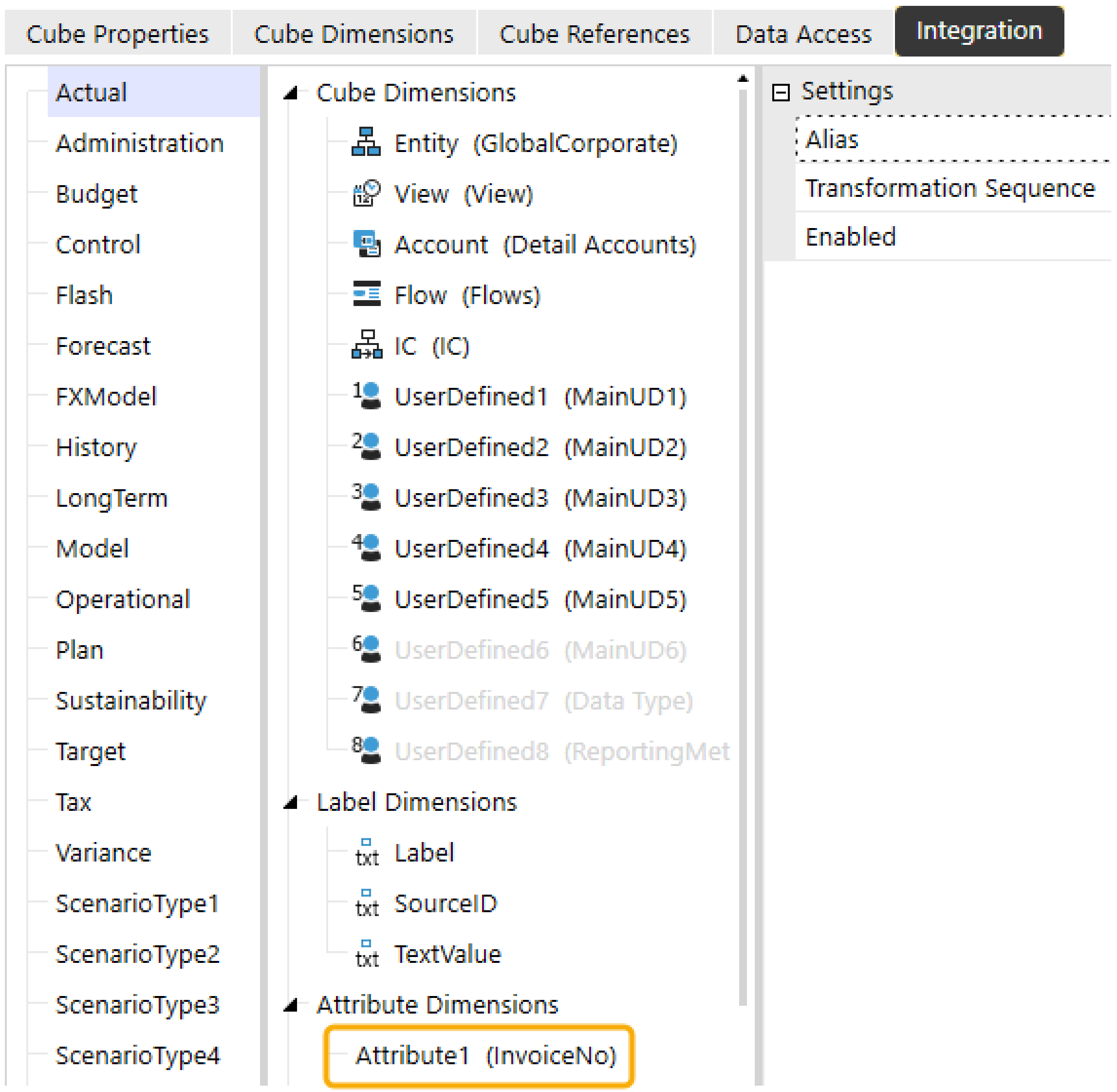
Data access and other data cell level controls are not supported in BI Blend.



Cube Properties
Cube Dimensions
Cube References
Data Access
Integration
Data Cell Access Security
Data Cell Conditional Input
Data Management Access Security

Integration

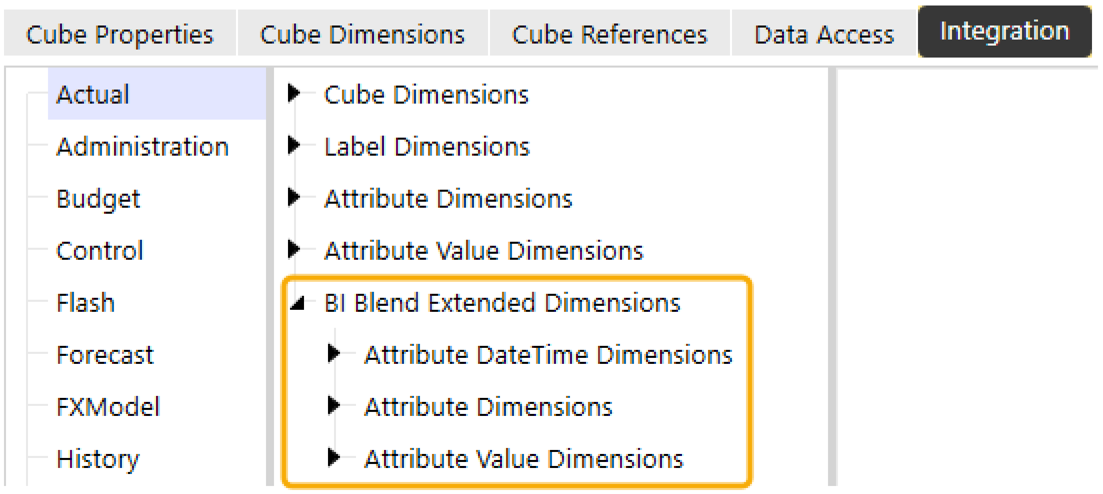
The integration for the cube provides the dimensionality available to the BI Blend Engine. The active dimensions can be used in both the transformed and un-transformed states, just like all data integration processes in the standard Stage processing. The additional 20 Attribute dimension members can be activated to support the inclusion of non-cube data in the BI Blend output. The Attribute Value dimension member can be activated and used in BI Blend to improve performance when loading large volumes of records containing many time periods by allowing all time to be associated by record.



BI Blend extended dimensions consist of three dimensions that provide an additional 57 fields to capture text, numeric values, and time:

- **Attribute DateTime Dimensions**: These 13 fields are designed to capture time-based attributes. The xBlendDateTime field usage is extended to function as the time-based property in Derivative Rule computational math functions

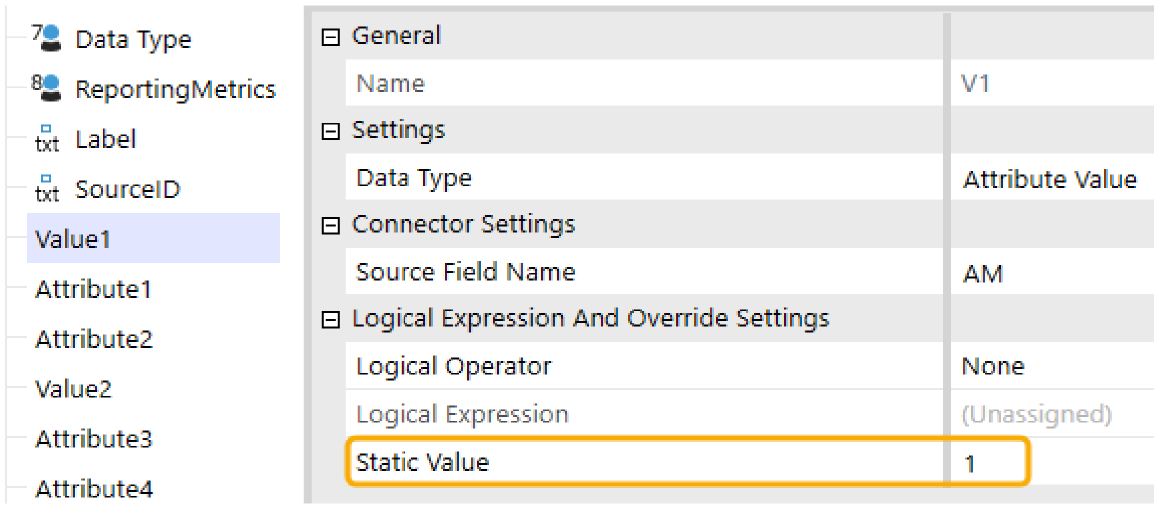
- **Attribute Dimensions**: These 20 text attributes expand the scope of the data set.
- **Attribute Value Dimensions**: These 24 additional attribute value dimensions extend the number of time periods that can be imported to 36.



Attribute Dimensions

External cube information is collected using the Attribute dimensions by enabling the fields in the cube Integration settings.

Attribute Value dimensions cannot be assigned static values when used with BI Blend workflows.



7 Data Type
8 ReportingMetrics
Label
SourceID
Value1
Attribute1
Attribute2
Value2
Attribute3
Attribute4

General	
Name	V1
Settings	
Data Type	Attribute Value
Connector Settings	
Source Field Name	AM
Logical Expression And Override Settings	
Logical Operator	None
Logical Expression	(Unassigned)
Static Value	1

Attribute Text Dimension

The cube Integration settings contain 20 Attribute Text dimensions available for data integration. By associating the imported members of an Attribute Text dimension to a UD8 metadata hierarchy, BI Blend can generate the aggregated parent level results for reporting. You can manage Attribute Dimensions in the following ways:

- Attributes as records can be aggregated within a Pivot Grid as a reporting tool feature.
- Aggregations of Attributes can be calculated using BI Blend Derivative Rules.
- Attributes can be associated with UD8 base members and aggregated within the selected UD8 dimension hierarchy.

Associating an Attribute with a UD8 base member requires the Attribute Dimension Record to exist as a base member in a UD8 Dimension. Only UD8 can be used to generate Attribute Dimension Members for aggregation. Because the record must be reflected as a dimension member, Aggregation Attributes may not be appropriate for highly transactional, changing, record members.

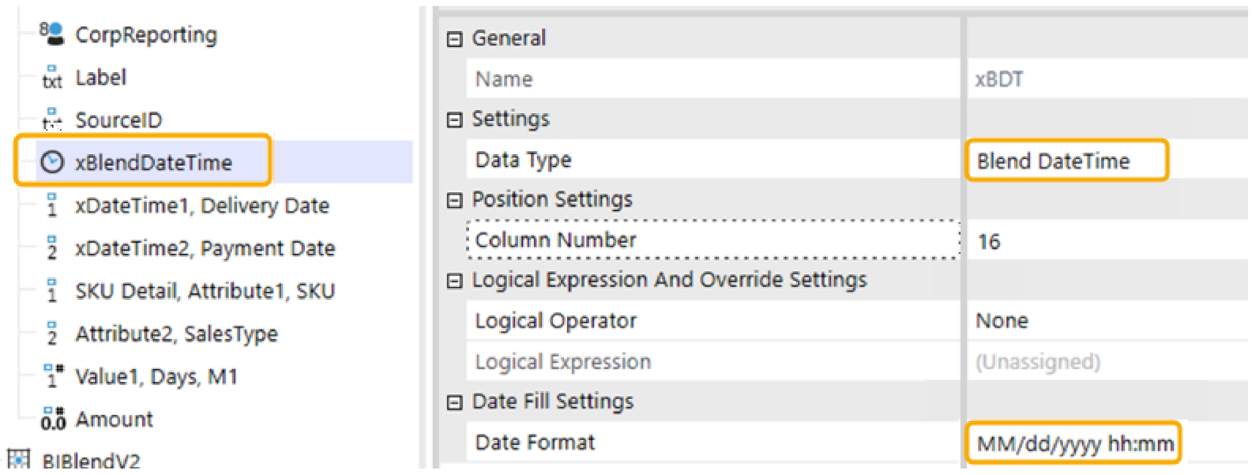
Attribute Value Dimension

Cube data integration provides 12 Attribute Value dimension members to capture numeric values. These fields capture Time field values for matrix data files.

Blend X Attribute DateTime Dimension

The Attribute DateTime dimension members are unique to BI Blend data sources and cannot be used on non-Blend workflow types. These fields are designed to capture Time based fields and have the extended capability to function as time values in derivative rule functions.

To use the DateTime dimension, the data source must have the integrated dimension set as a Blend DateTime data type. The formatting as MM/dd/yy hh:mm is required to support the time-based derivative rule functions.



- CorpReporting
- Label
- SourceID
- xBlendDateTime
- xDateTime1, Delivery Date
- xDateTime2, Payment Date
- SKU Detail, Attribute1, SKU
- Attribute2, SalesType
- Value1, Days, M1
- Amount
- BIBlendV2

General	
Name	xBDT
Settings	
Data Type	Blend DateTime
Position Settings	
Column Number	16
Logical Expression And Override Settings	
Logical Operator	None
Logical Expression	(Unassigned)
Date Fill Settings	
Date Format	MM/dd/yyyy hh:mm

Settings include:

- **xBlendDateTime**: This is a special column field used by the BI Blend engine to serve derivative calculations for DataSeries/Time value computations.
- **xBlendTime**: Twelve time column fields to collect additional time calculations that can support date difference calculations.

Blend X Attribute Text Dimension

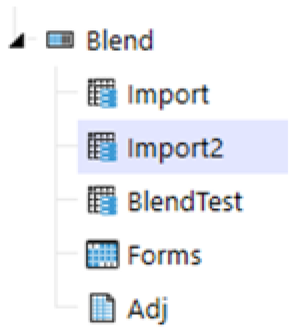
Twenty additional text column fields are unique to Blend workflows. These fields are not supported in other non-Blend workflows. Additionally, the Blend X Attribute Text fields do not support extending the records to UD8 for parent level aggregations.

Blend X Attribute Value Dimension

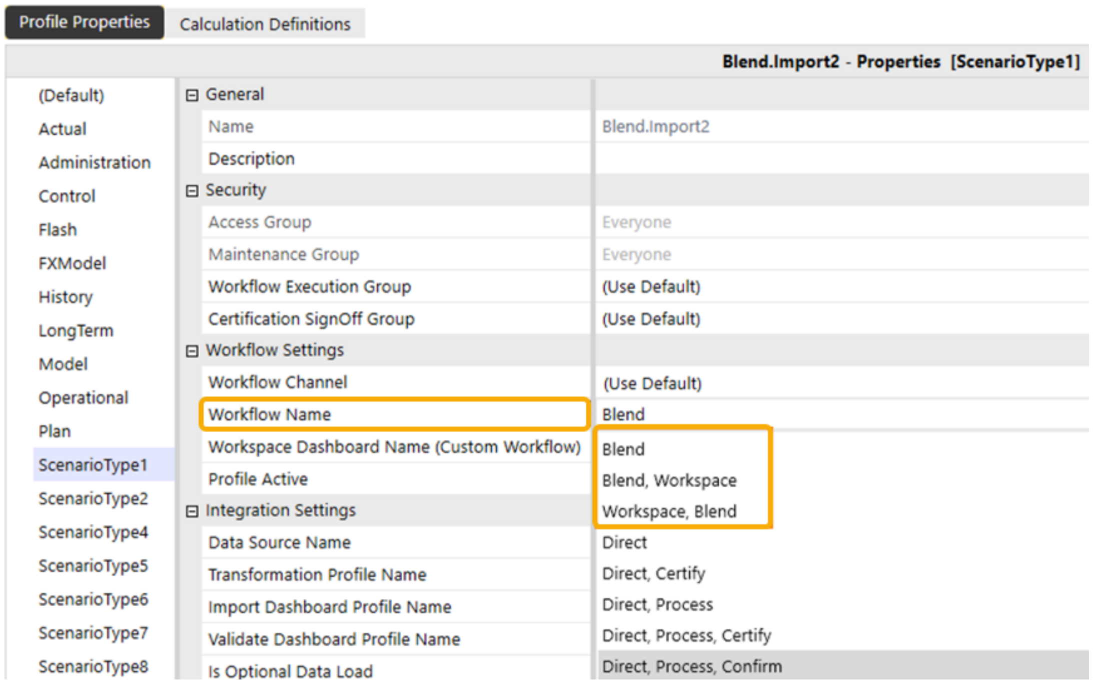
Twenty-four additional value column fields are unique to Blend workflows and are used to capture numeric values. These fields are not supported in other non-Blend workflows. The Attribute Value fields can be used to map time to create a matrix load. The combination of Attribute Value and Blend X Attribute Value fields provides a maximum time-based matrix load of 36 periods.

Set Up Workflow for BI Blend

1. Create a Workflow structure to manage BI Blend. Administrators apply the appropriate Security groups as required.



2. By Scenario Type, select the import channel to Blend.
    a. **Blend**: Limited to the BI Blend data processing.
    b. **Blend, Workspace**: Additional support to display Workspace Dashboards.

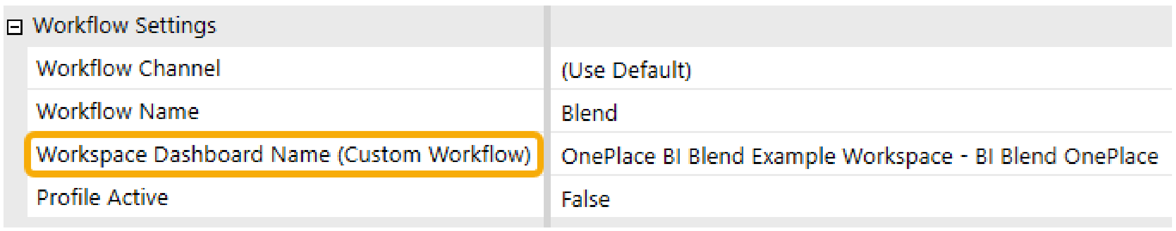


Blend.Import2 - Properties [ScenarioType1]

(Default), Actual, Administration, Control, Flash, FXModel, History, LongTerm, Model, Operational, Plan, ScenarioType1, ScenarioType2, ScenarioType4, ScenarioType5, ScenarioType6, ScenarioType7, ScenarioType8

⊟ General	
Name	Blend.Import2
Description	
⊟ Security	
Access Group	Everyone
Maintenance Group	Everyone
Workflow Execution Group	(Use Default)
Certification SignOff Group	(Use Default)
⊟ Workflow Settings	
Workflow Channel	(Use Default)
Workflow Name	Blend
Workspace Dashboard Name (Custom Workflow)	Blend
Profile Active	Blend, Workspace
⊟ Integration Settings	Workspace, Blend
Data Source Name	Direct
Transformation Profile Name	Direct, Certify
Import Dashboard Profile Name	Direct, Process
Validate Dashboard Profile Name	Direct, Process, Certify
Is Optional Data Load	Direct, Process, Confirm

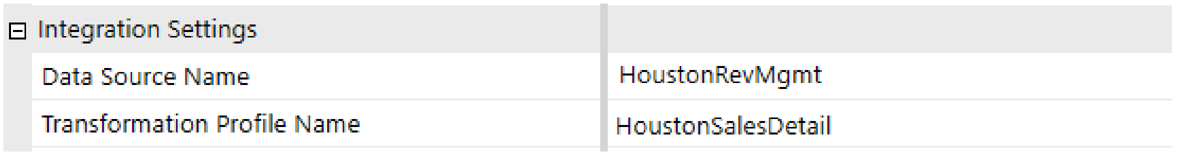
3. If designing BI Blend as a Workflow Process, dashboards are assigned using the Workspace Dashboard Name in the Workflow Settings.



⊟ Workflow Settings	
Workflow Channel	(Use Default)
Workflow Name	Blend
Workspace Dashboard Name (Custom Workflow)	OnePlace BI Blend Example Workspace - BI Blend OnePlace
Profile Active	False

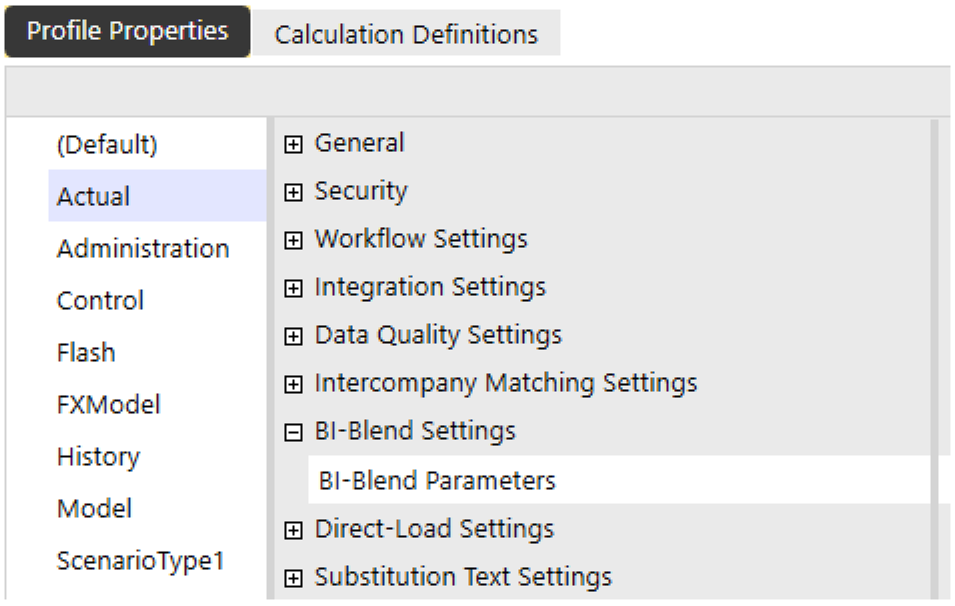
4. Assign the Data Source and Transformation Rule Profile which BI Blend will use to import

and transform data.



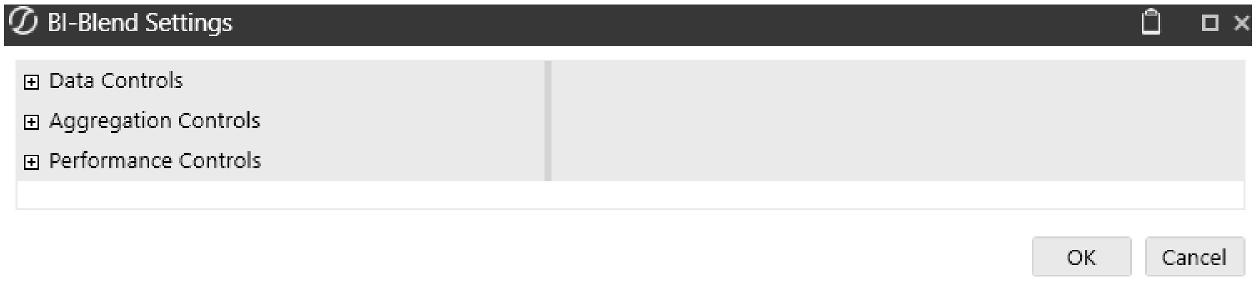
⊟ Integration Settings	
Data Source Name	HoustonRevMgmt
Transformation Profile Name	HoustonSalesDetail

5. Set the BI Blend Parameters which are a collection of properties that define the BI Blend process.

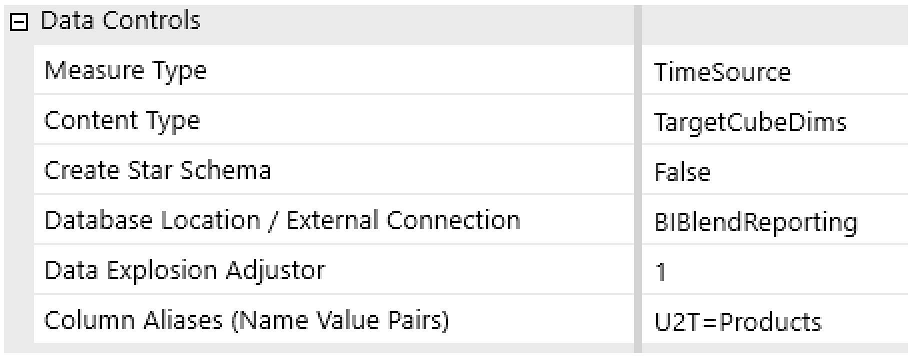


Workflow Profile BI-Blend Settings

A primary design element of BI Blend is the BI-Blend Settings available in the workflow profile Profile Properties by Scenario Type. These settings enable the BI Blend administrator to define and optimize the generation of the BI Blend tables and views to meet reporting requirements. They also enable administrators to define the desired aggregation points and enable performance control updates.



Data Controls Settings



Data Controls	
Measure Type	TimeSource
Content Type	TargetCubeDims
Create Star Schema	False
Database Location / External Connection	BIBlendReporting
Data Explosion Adjustor	1
Column Aliases (Name Value Pairs)	U2T=Products

Measure Type

Defines how the time dimension column will be determined in the relational tables. You can choose from the following options:

- **Time Source**: The time/amount columns generated in the Blend table are determined by the members in the source data set.
- **TimeWFView**: The time/amount columns generated in the Blend table are determined by the Workflow Tracking and Input Frequency set for the workflow scenario. For example, a yearly tracking frequency for a monthly input scenario would create 12 time columns regardless of the number of periods in the data records. This option is used for Time Math helper rules.

- **TimeWFViewAV**: Used for matrix loads using Attribute Value and BI Blend Extended Attribute dimensions only. The time/amount columns generated in the Blend table are determined by the Workflow Tracking and Input Frequency set for the workflow scenario. This solution requires time based value records to be associated with Attribute Value Dimension members in the data source. Each value (1-36) is associated with a time period column in the output table. This method efficiently processes records in large multi-period datasets.

Content Type
Determines the record detail that is written to the BI Blend tables. Cube dimensions, such as Entity, Account, and UD, are supported by Source and Target, or transformed, results. You can also include or exclude the attribute or BI Blend Extended Dimensions. The analysis of source against aggregated parents is not supported. You can choose from the following options:

- **TargetCubeDims**: Only transformed cube dimension target members columns are imported.
- **TargetCubeDimsSource**: Source and transformed cube dimension target member columns are imported.
- **TargetCubeDimsAttributes**: Transformed cube target members and attribute members are imported. For example, when data is integrated to Attribute dimensions for non-cube dimension records, such as invoices assigned as an attribute.
- **TargetCubeDimsAll**: Source, target, and attribute members are imported.
- **TargetCubeDimsAttributeEx**: Transformed cube dimension target members, attribute members, and BI Blend Extended members are imported.
- **TargetCubeDimsAllEx**: Source, target, attribute, and BI Blend Extended members are imported.

Create Star Schema

If set to True, a view is created during import based on the aggregation dimension tables. The BI Blend database is generated as a Star Schema table set with related views, which is required for Leveled Hierarchy reporting.

Supporting dimension tables are created for each cube dimension assigned an Aggregation Control. The Dimension tables will contain column fields for MemberName, MemberDesc, NameAndDesc, ParentName, IndentLevel, IsBase, and MemberSeq, which can be used in reporting against the Star Schema view. This is an optional setting.

Database Location / External Connection

The connection to the storage location for the Blend tables. Select the connection name.

Data Explosion Adjuster

The integer value entered controls data explosion and estimates the size of tables generated, as derived from the initial Learning Mode. The default value is one.

Colum Aliases (Name Value Pairs)

This setting enables you to use column aliases to define custom dimension column names in the main Blend table using the stage table keywords. If left blank, the system generates column names based on the default table and dimension labels. This will override aliases set on the cube Integration tab.

> **Example:** V1=OrderQuantity, V2=UnitPrice

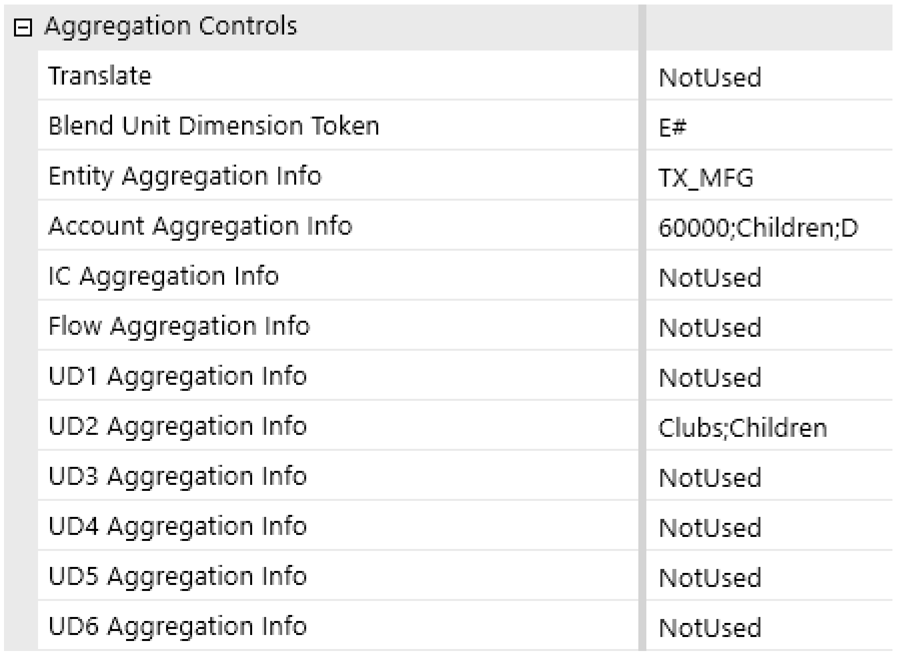
> **Example:** This example is for custom label columns for UD dimensions and Attribute Text dimensions: U1T=CostCenter,U2T=Products,U3T=Regions,A1=DepartmentCode,A2=ExpenseID.

Review these column alias keys:

Dimension	Column Alias Key
UD1-UD8	U1T-U8T
Attribute Text Dimension	A1-A20
Attribute Value Dimension	V1-V12
Label	Lbl
TextValue	Tv
Blend X Attribute	xA1-xA40
Blend X Attribute Value	xV13-xV36

NOTE: Alias must be unique. Multiple aliases can be set, separated by a comma. You cannot have spaces in your alias value names, and quotation marks are not required.

Aggregation Controls Settings



Aggregation Controls	
Translate	NotUsed
Blend Unit Dimension Token	E#
Entity Aggregation Info	TX_MFG
Account Aggregation Info	60000;Children;D
IC Aggregation Info	NotUsed
Flow Aggregation Info	NotUsed
UD1 Aggregation Info	NotUsed
UD2 Aggregation Info	Clubs;Children
UD3 Aggregation Info	NotUsed
UD4 Aggregation Info	NotUsed
UD5 Aggregation Info	NotUsed
UD6 Aggregation Info	NotUsed

Translation
Enables translation to a single destination currency. Translation occurs during a Blend import, even if Entity and Account do not have aggregation info defined. Only the direct method is applied based on the rate types defined in the cube settings, which are determined by the account type. No complex currency translation is supported by sub-parent levels. All Entities are translated to the destination currency. You can choose from the following options:

- **Not Used**: Amounts are not translated
- **Currency symbol**: Amounts are translated to the selected currency

Blend Unit Dimension Token
An aggregation point must be set for the Blend Unit dimension. This assigns the cube dimension as the partitioning dimension to generate pages and the corresponding level of multi-threading. These are the available dimensions for the Blend Unit:

- **MaxMembersDim**: Dynamically sets the Blend Unit as the dimension that has an aggregation point set and has the most members
- **E#**: Sets Entity as the Blend Unit
- **F#**: Sets Flow as the Blend Unit
- **UD1#-UD8#**: Sets the chosen UD dimension as the Blend Unit
- **A1#-A20#**: Sets the chosen Attribute dimension as the Blend Unit

Aggregation Controls

Dimension/Attribute Aggregation Info

These properties are used to set the parent level for the top-level aggregation. These filters limit the return of records to the parents that need aggregation for reporting. The complete data set is used, but records outside any aggregation path are ignored. This enables BI Blend to generate results focused on specific reporting and analysis requirements.

> **Example:** If an aggregation control for Entity is set for US Clubs, any records present in the data set for Entities outside the US Clubs hierarchy are ignored.

When using BI Blend with extended dimensions, BI Blend only aggregates the ultimate base members of an extended dimension. Loading a member that becomes a parent as a result of dimension inheritance, such as mapping source data to a parent member, results in no aggregation for the extended parent.

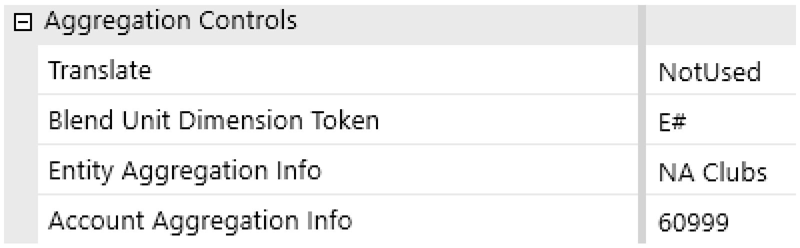
Aggregation controls can be set on any dimension or attribute in addition to the Blend Unit dimension. Setting additional aggregation points on non-Blend Unit dimensions and attributes results in base-level records being aggregated up those hierarchies in addition to the Blend Unit hierarchy.

Aggregation Control Types

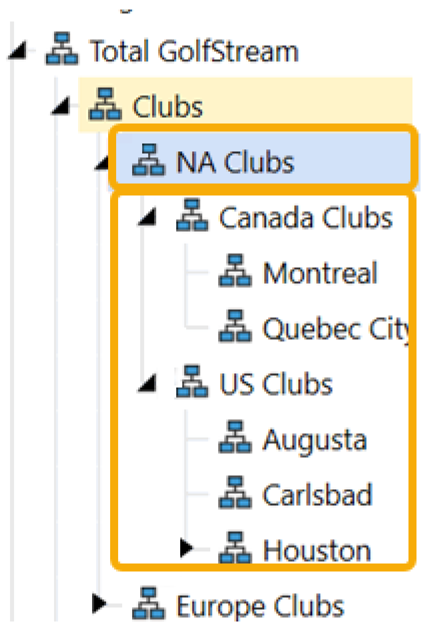
If NotUsed is selected, the records import to the base-level members only for that dimension. The records will be collected at the row record level if included in the data source.

For parent level selection, selecting a single parent limits the records to the members within that hierarchy and generates records for all members with data within the hierarchy.

> **Example:** Selecting NA clubs would create records for NA Clubs as TreeDescendentsInclusive.



Aggregation Controls	
Translate	NotUsed
Blend Unit Dimension Token	E#
Entity Aggregation Info	NA Clubs
Account Aggregation Info	60999



Aggregation, label, and Star Schema options are applied to the import in accordance with the controls set. They are applied in this order:
TopMember;RestrictMember;Labels;StarSchemaControl.

If you are using a UD8 dimension, you must include the dimension hierarchy name.

> **Example:**
> DimensionName:TopMember;RestrictMember;Labels;StarSc
> hemaControl.

If you are not using an option, you still need to include the semicolon as a placeholder for the option.

> **Example:** TopMember;;;StarSchemaControl.

> **NOTE:** Applying aggregation information filters returns both the filtered parents and the base member records. The exception to this behavior is when the Blend Unit dimension uses the ;Member filter. The Member filter on a Blend Unit dimension returns only that member.

Member Control

Parent Level Selection, as Member
The dimension assigned as the Blend Unit can be restricted to return a summary parent member. In this situation, the results are only required for an aggregated parent. By entering the setting as NA Club;Member, all the descendant's results will be aggregated to the parent, but the descendant members will not be included in the output. The option for a specified Parent Member is only available in the Blend Unit.

For non-Blend Unit dimensions, using ;Member will return the Parent Member defined, as well as all the base members found within the parent member's hierarchy.

Parent Level Selection, Children
This focuses aggregation points to a member's children. The selection of NA Clubs;Member.Children property returns the children of the parent member as Children Inclusive. Alternatively, NA Clubs;Children can be used as a non-inclusive filter. Base members are also returned.

Parent Level Selection, TreeDescendants
This is a non-inclusive option. Records are aggregated and returned for every level in the hierarchy except for the named member.

Dimension Syntax	Example	Impact
Member		
<Member name>;Member	Clubs;Member	**Blend Unit dimension**: All records are aggregated on the named member, but only records for the named member are returned and not any base-level records. **Other dimensions**: Base-level records are returned in addition to the records aggregated on the named member.
Children		

Dimension Syntax	Example	Impact
<Member name>;Children	Clubs;Children	Records are aggregated on the named member's children but not on the named member.
Member.Children		
<Member name>;Member.Children	Clubs;Member.Children	Records are returned for the named member's children and the base-level members.
Tree Descendants		
<Member name>;TreeDescendants	Clubs;TreeDescendants	Records are aggregated and returned for every level in the hierarchy except for the named member.
No Expansion		
<Member name>	Clubs	Records are aggregated and returned for every level in the hierarchy including the named member

Label Control

Reporting Labels as Name, Description or Name and Description

By default, any dimension set as an aggregation will return the results using the Name field found in the dimension properties. To modify results, the dimension or attribute must be used as an aggregation control. Using the label properties on a standard non-Star Schema BI Blend table replaces the record with the labeling method.

Dimension Syntax	Example	Impact
N: Name		
Member name>;;N	Clubs;;N	The member name is used in the imported records. This is the default setting.
D: Description		
<Member name>;;D	Clubs;;D	The member description is used in the imported records.
ND: Name and Description		
<Member name>;;ND	Clubs;;ND	Both the member name and description are used in the imported records.

Star Schema Control

Star Schema Only

The SSOnly filter can be used when the Create Star Schema property is set to True in the Data Controls settings. This setting will not generate parent level records in the BI Blend data table. The SSOnly aggregation control creates a complete Star-Schema dimension table containing the following fields for cube dimension hierarchy as specified by the parent member in the aggregation control: MemberName, MemberDesc, NameAndDesc, ParentName, IndentLevel, IsBase, and MemberSeq .

Star Schema Leveled Hierarchy

The SSLeveled property can be used when the Create Star Schema property is set to True in the Data Controls settings. If enabled, the corresponding Star-Schema dimension table has to have zero-based column fields added, which corresponds to the hierarchical structure of the dimension. This is not valid for use on Account and Attributes. See Leveled Hierarchy.

NOTE: An aggregation point or member must be defined on the Blend Unit dimension.

Dimension Syntax	Example	Impact
SSOnly		
<Member name>;SSOnly	Clubs;SSOnly	No records are aggregated during the import. A dimension table is produced to capture the dimension hierarchy. This control cannot be used with a member or label control. This can only be used if Star Schema is enabled.
SSLeveled		

Dimension Syntax	Example	Impact
<Member name>;;;SSLeveled	Clubs;;;SSLeveled	Additional hierarchy description columns are added to the dimension tables and view. Star Schema must be enabled.

Combining Controls

All controls specified are applied. Any combination of control types can be used. They are applied in this order: <Member name>;<Member expansion>;<Label option>;<Star Schema control>.

> **Example:** Clubs;Member;ND;SSLeveled

> **Example:** Houstin;Member;ND

> **Example:** Houstin;Member.Children;D

> **Example:** Houstin;;;SSLeveled

Attribute Dimension Syntax

Prefix the dimension syntax expression with the <UD8 dimension hierarchy name>;. The number of fields is larger due to the inclusion of the dimension name.

> **Example:** Dimension Name; TopMember;RestrictMember;Labels

> **Example:** UD8BlendAttributes;Contracts;;D

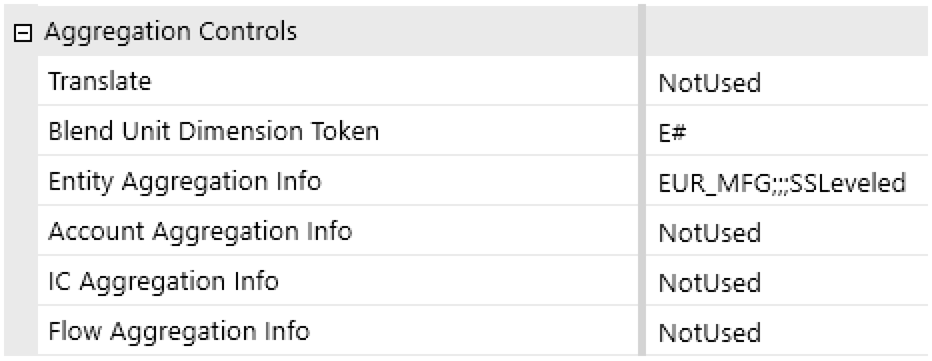
> **Example:** UD8BlendAttributes;Contracts;Member;D

Leveled Hierarchy

Leveled hierarchy is for aggregation reporting where parent values reflect the aggregation points. It is activated as a BI-Blend aggregation control impacting the Star-Schema dimension tables with the creation of Leveled and IsBaseBIBlend column fields. Leveling adds the hierarchy context that is useful in Pivot Grids and custom dashboard reports.

Use the following syntax: TopMember;RestrictMember;Labels;SSLeveled

Example: EUR_MFG;;;SSLeveled



⊟ Aggregation Controls	
Translate	NotUsed
Blend Unit Dimension Token	E#
Entity Aggregation Info	EUR_MFG;;;SSLeveled
Account Aggregation Info	NotUsed
IC Aggregation Info	NotUsed
Flow Aggregation Info	NotUsed

The column fields generated by the SSLeveled property are created to the maximum depth of the hierarchy, starting from the defined parent to the base member. The leveled columns are only created on base members and their ancestors where data is populated in the BI-Blend data table. Levels greater than those containing data are not created.

NOTE: The property is not valid for use on Account and Attribute aggregation controls.

The leveling process generates two field placeholders:

- **XFLeveled** : Created only on base-level records for data intersections whose hierarchy is less than the maximum depth.
- **XFStored** : Created only on parent-level members. This represents the data intersection of stored-parent values created by the BI-Blend engine

Leveled Hierarchy Processing

The SSLeveled property enables the generation of the hierarchy leveling fields. Efficiency is accomplished by limiting the leveling only to members that contain data in the BI-Blend data table. However, leveling is an additional process used by the BI-Blend Engine which will affect the overall BI-Blend processing time.

Star-Schema Leveling Column Fields

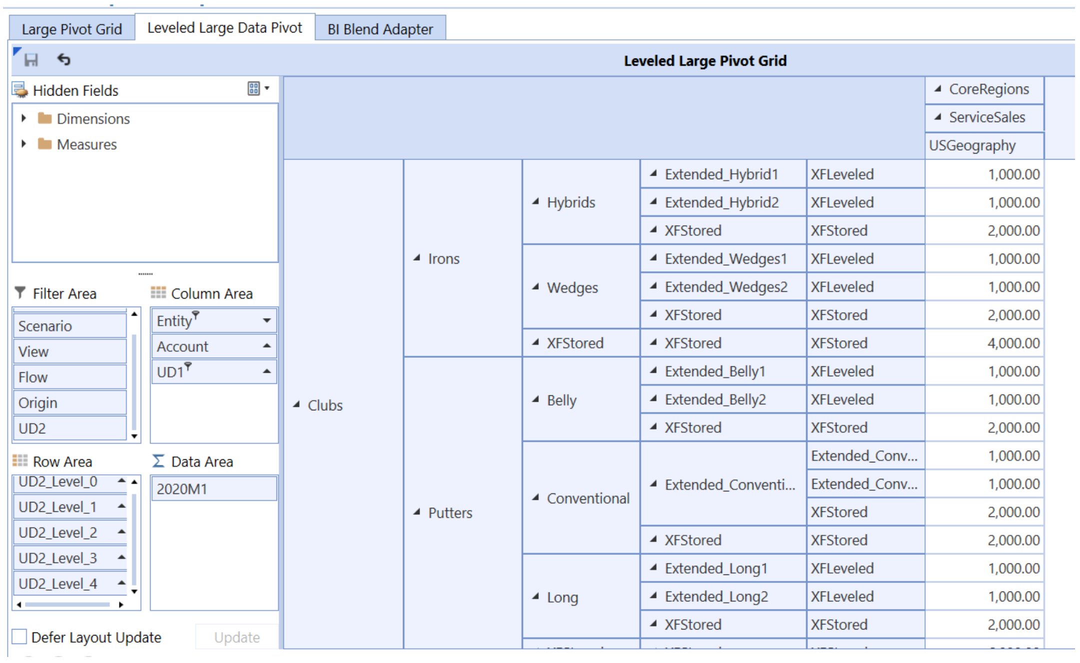
The SSLeveled property enables the generation of new columns in corresponding Star-Schema dimension tables. BI Blend will only aggregate the ultimate base members of an extended dimension. Loading a member that becomes a parent as a result of dimension inheritance will result in no aggregation for the extended parent.

- **IsBIBlendBase**: Reflects the hierarchy status of where the actual data records exist. Restricts the creation of leveling only on members that contain data in the data table. Designates the member as a base (1) or parent member (0) by the usage in the BI-Blend data table.
- **Level x**: Zero-based to maximum descendent depth only on members where BI Blend data exists.

NOTE: The BI Blend leveled hierarchy is only available for Star-Schema-enabled workflows.

Leveled Hierarchy in a Large Data Pivot Grid

This example uses the leveled hierarchy in the Large Data Pivot Grid. The table has a hierarchical view.



Filter Area: Scenario, View, Flow, Origin, UD2

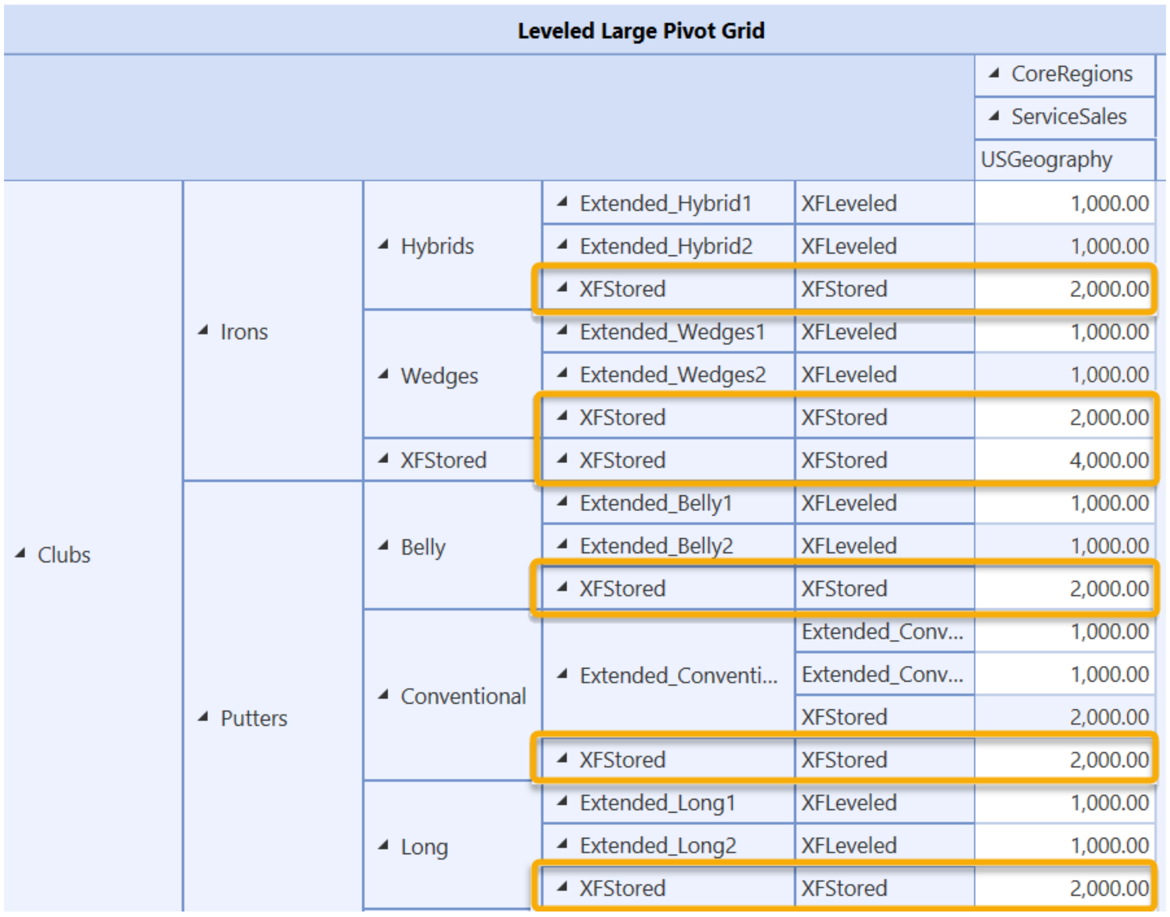
Column Area: Entity, Account, UD1

Row Area: UD2_Level_0, UD2_Level_1, UD2_Level_2, UD2_Level_3, UD2_Level_4

Data Area: 2020M1

					CoreRegions
					ServiceSales
					USGeography
Clubs	Irons	Hybrids	Extended_Hybrid1	XFLeveled	1,000.00
			Extended_Hybrid2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00
		Wedges	Extended_Wedges1	XFLeveled	1,000.00
			Extended_Wedges2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00
		XFStored	XFStored	XFStored	4,000.00
	Putters	Belly	Extended_Belly1	XFLeveled	1,000.00
			Extended_Belly2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00
		Conventional	Extended_Conventi...	Extended_Conv...	1,000.00
				Extended_Conv...	1,000.00
				XFStored	2,000.00
			XFStored	XFStored	2,000.00
		Long	Extended_Long1	XFLeveled	1,000.00
			Extended_Long2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00

Notice in this example that the items labeled XFStored indicate that the table is being leveled to create leveled columns based on the maximum depth of hierarchy from the BI Blend data.



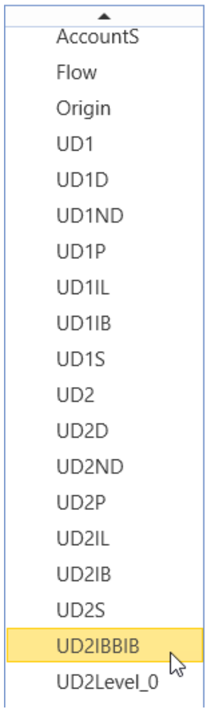
Leveled Large Pivot Grid					
					CoreRegions
					ServiceSales
					USGeography
Clubs	Irons	Hybrids	Extended_Hybrid1	XFLeveled	1,000.00
			Extended_Hybrid2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00
		Wedges	Extended_Wedges1	XFLeveled	1,000.00
			Extended_Wedges2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00
		XFStored	XFStored	XFStored	4,000.00
	Putters	Belly	Extended_Belly1	XFLeveled	1,000.00
			Extended_Belly2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00
		Conventional	Extended_Conventi...	Extended_Conv...	1,000.00
				Extended_Conv...	1,000.00
				XFStored	2,000.00
			XFStored	XFStored	2,000.00
		Long	Extended_Long1	XFLeveled	1,000.00
			Extended_Long2	XFLeveled	1,000.00
			XFStored	XFStored	2,000.00

Setting IsBIBlendBase

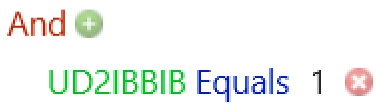
You can use the IsBaseBIBlend parameter to show where actual data records exist in the table. Complete the following steps:

1. In the leveled Large Pivot Grid, right-click in the row and select **Show Prefilter** to display the **PivotGrid Prefilter** dialog box.
2. Click the **+**.

3. Click **Rt** and scroll through the drop-down menu to select **UD2IBBIB**.

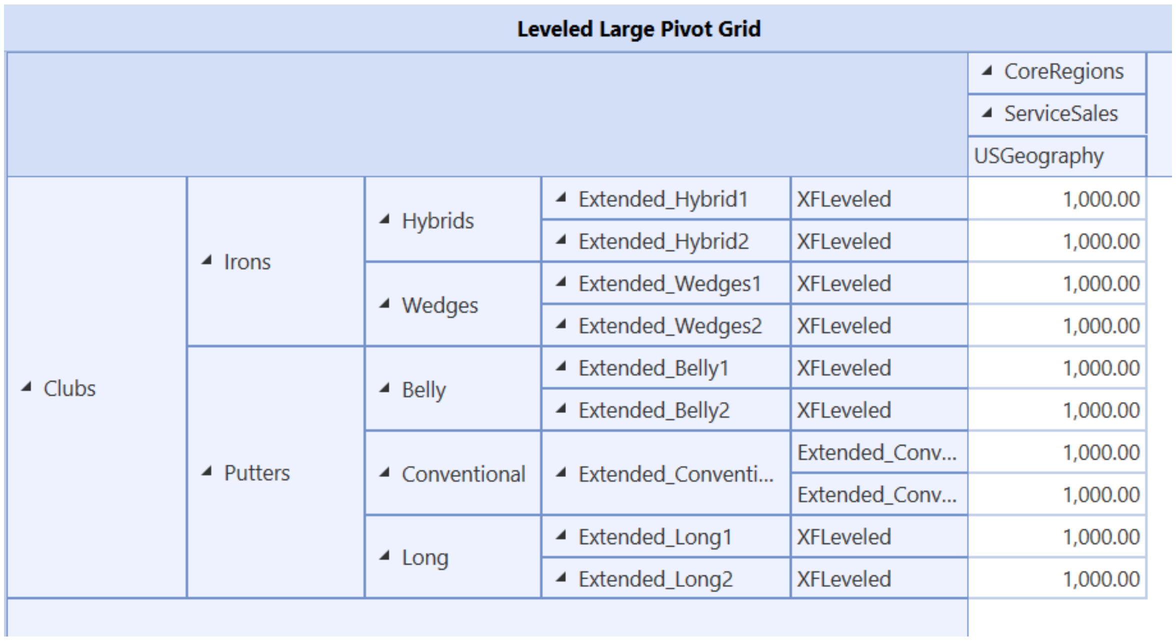


4. Click **<enter a value>** and enter **1**. The equation is UD21BBIB Equals 1.



5. Click the **OK** button. After applying IsBaseBIBlend, the leveled table now changes to show only the rows that contain actual data (XFLeveled), and it has removed the rows that were

added for leveling (XFStored).

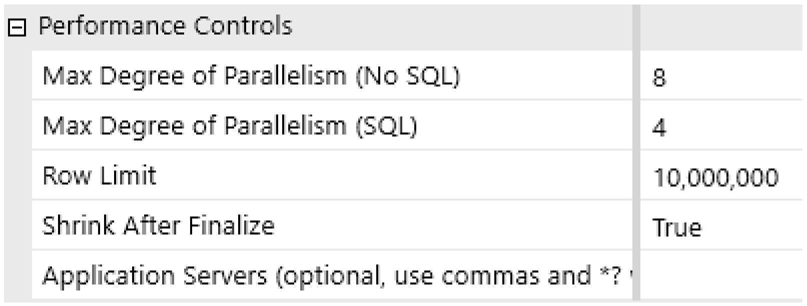


Leveled Large Pivot Grid

					CoreRegions
					ServiceSales
					USGeography
Clubs	Irons	Hybrids	Extended_Hybrid1	XFLeveled	1,000.00
			Extended_Hybrid2	XFLeveled	1,000.00
		Wedges	Extended_Wedges1	XFLeveled	1,000.00
			Extended_Wedges2	XFLeveled	1,000.00
	Putters	Belly	Extended_Belly1	XFLeveled	1,000.00
			Extended_Belly2	XFLeveled	1,000.00
		Conventional	Extended_Conventi...	Extended_Conv...	1,000.00
				Extended_Conv...	1,000.00
		Long	Extended_Long1	XFLeveled	1,000.00
			Extended_Long2	XFLeveled	1,000.00

8. Click the **Save** button to save your changes.

Performance Controls Settings



Performance Controls	
Max Degree of Parallelism (No SQL)	8
Max Degree of Parallelism (SQL)	4
Row Limit	10,000,000
Shrink After Finalize	True
Application Servers (optional, use commas and *?	

Max Degree of Parallelism (No SQL)

The work of importing data can be split between multiple parallel processors. The default setting is 8 for non-SQL operations (Application Server). The optimal number depends on your server configuration.

IMPORTANT: The default value should only be adjusted with guidance from Product Support or your implementation consultant.

Max Degree of Parallelism (SQL)

This option controls how many threads are used to perform parallel tasks on the SQL database server itself, such as writing to tables. The default value is 4.

IMPORTANT: The default value should only be adjusted with guidance from Product Support or your implementation consultant.

Row Limit

This sets an upper limit on the number of records written to the Blend table per execution of query. The default is set to 100K. The maximum is 500K. This is estimated by analyzing the Blend Log File, base level rows, and the Explosion factor to determine free memory usage.

Shrink After Finalize

When set to True, this setting reduces the amount of empty space in the database. This property should not be changed.

Application Servers

Used to identify which servers to use to process the Blend import and aggregation. The BI Blend processes default to the Stage Servers with queuing controls. This can be overridden with named servers.

NOTE: This property is only used for Platform Version 7.4.4 and earlier. Do not name application servers using this field in Platform Versions 8.0.0 and later.

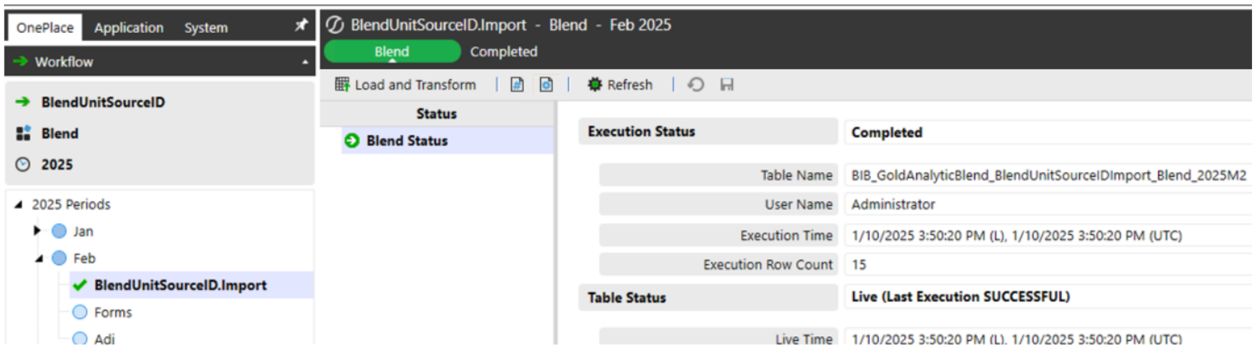
Import Blend Data

The workflow name used can be Blend, Blend Workspace, or Workspace Blend. Your import options are the same as other OneStream Workflows that commonly result in cube data.

The configuration of BI Blend is managed in the Workflow Profile. However, some of the Stage Engine's workflow properties may not be valid or used in BI Blend configurations. The primary configuration for BI Blend is done using the workflow name and BI Blend settings. The BI Blend Engine's architecture, such as in-memory processing, makes some workflow properties, such as the Append Default Load Method, an invalid selection. Such selections do not have an effect on workflow blend behaviors.

Data can be collected using:

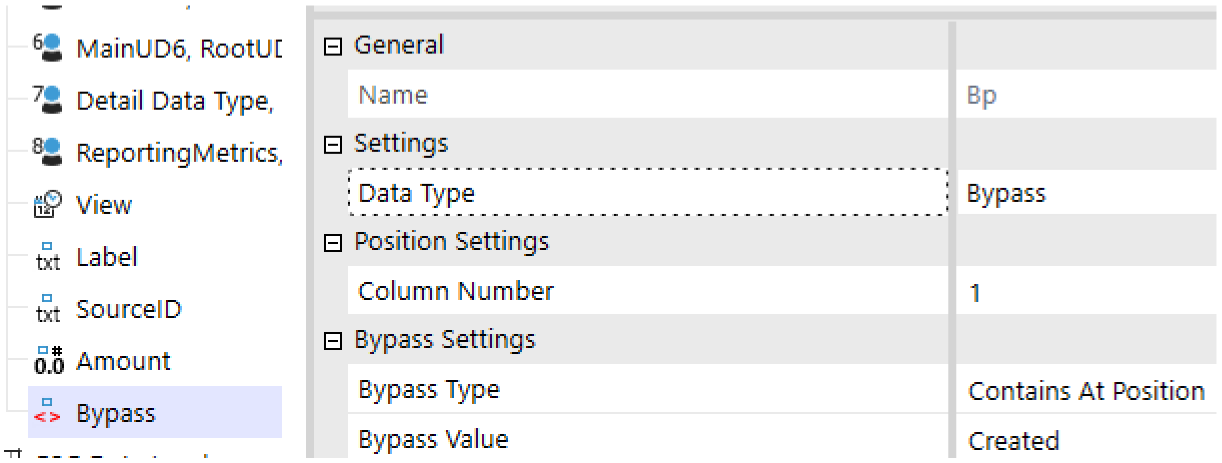
- Files (Fixed or Delimited)
- Connectors
- Other workflow Stage data



Execution Status	Completed
Table Name	BIB_GoldAnalyticBlend_BlendUnitSourceIDImport_Blend_2025M2
User Name	Administrator
Execution Time	1/10/2025 3:50:20 PM (L), 1/10/2025 3:50:20 PM (UTC)
Execution Row Count	15
Table Status	Live (Last Execution SUCCESSFUL)
Live Time	1/10/2025 3:50:20 PM (L), 1/10/2025 3:50:20 PM (UTC)

Delimited Files

When loading data to a BI Blend workflow using a delimited file, a bypass must be used if the first row of the file contains column headers.



⊟ General	
Name	Bp
⊟ Settings	
Data Type	Bypass
⊟ Position Settings	
Column Number	1
⊟ Bypass Settings	
Bypass Type	Contains At Position
Bypass Value	Created

How to Use a File Bypass

Use a bypass to prevent file load failures in cases where a delimited file contains column headers.

1. Create a bypass:
   a. Navigate to **Application** > **Data Collection**>**Data Sources**.
   b. Under **Delimited Files**, select a data source.
   c. Click **Create Source Dimension**.
   d. From the **Name** drop-down menu, select **Bypass**.
   e. Click the **OK** button. The default Bypass Type is Contains Within Line.
   f. In the **Bypass Value** field, type the name of the first column header.
   g. Click the **Save** button.
2. Load the delimited file into a BI Blend workflow.

Static Values in Fields

In Platform Version 7.4.4 to Platform Version 8.1.3, field values are bypassed if a Static Value is applied to the field. In Platform Version 8.2 and later, fields with a Static Value are no longer bypassed.

To replicate this functionality and drop certain lines, typically headers, create a Complex Expression or Business Rule that bypasses the field.

For example, the following rule, applied to the Account field, would ensure the line containing the Account header is ignored:

```
'Check the parsed value for the criteria string
If StringHelper.FindIgnoreCase(args.Value, "Account") Then
    'Set the bypass switch on the globals object in order to tell the parser engine to skip
this line.
    globals.Bypass = True
End If
Return args.Value
```

Blend Steps

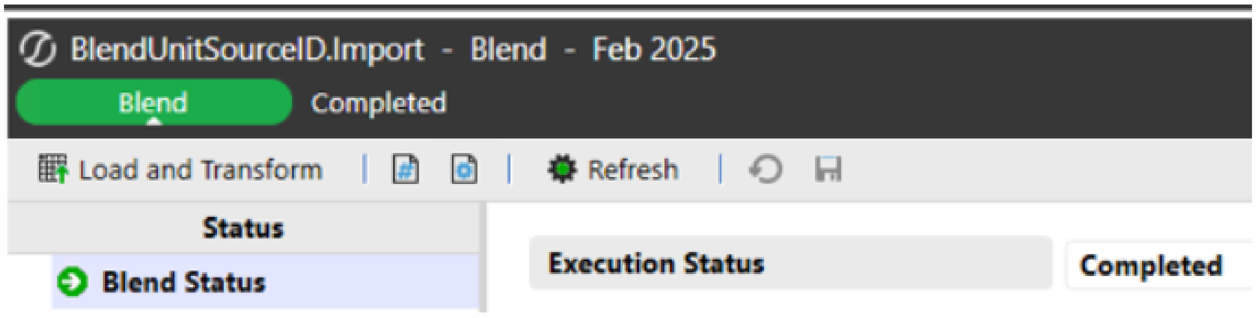
1. Data is queued for processing on an Application server.
2. Cube dimensions are integrated.
3. Derivative Rules and Transformation Rules are executed.
4. Blend Derivative Rules are executed
5. BI Blend Engine is initialized
6. BI Blend Transformation Rules are validated
7. Multi-threading of Page dimensions or Blend Units begins
8. BI Blend tables and view are dropped, recreated, and written to.

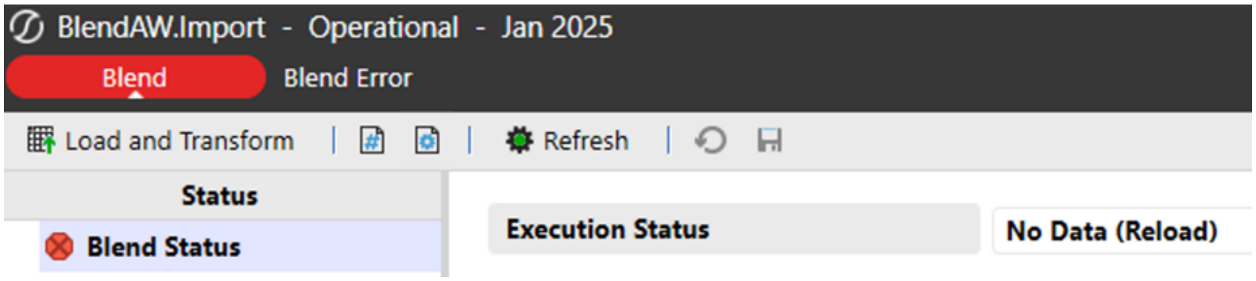
Validating Members

BI Blend processes the records in relation to the cube dimensions and the assigned Transformation Rules to create a common reporting solution. If any member in a record does not find a target in the cube dimension through Transformation Rules, a mapping error is presented. Validation is limited to target mapping. Validation of data intersections is not performed, such as those from constraint properties.

BI Blend Status

The BI Blend Status page displays information for the current Blend task. This indicates what results are available. If there is a failure in the Blend task, the correction must be performed and the entire task re-executed.



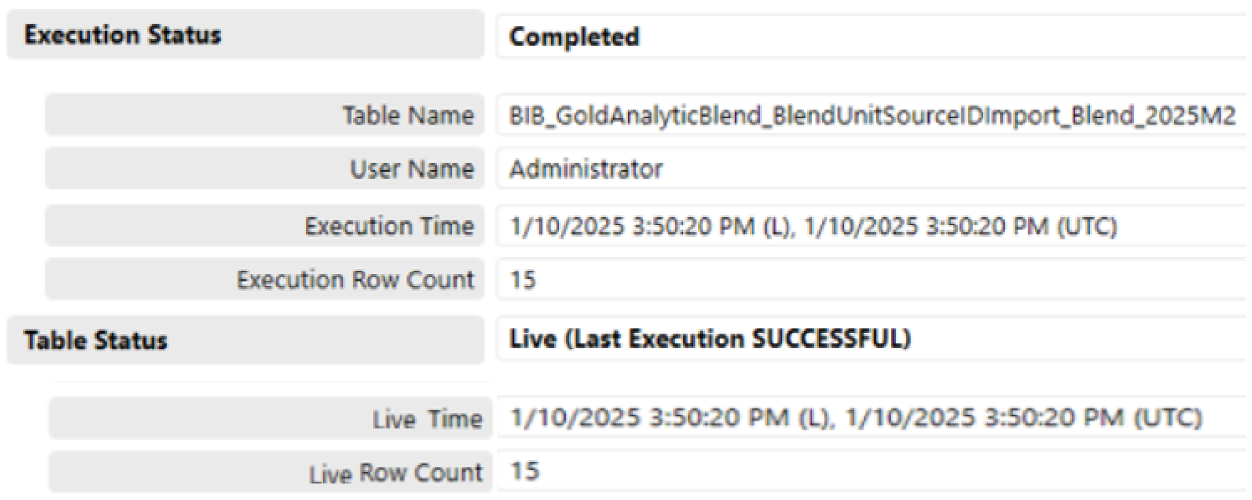


Execution Status

Displays the results of the most current Blend task. It notes information such as target table name and time. If errors were encountered, the Reload status displays to indicate the Blend task must be re-executed.

Table Status

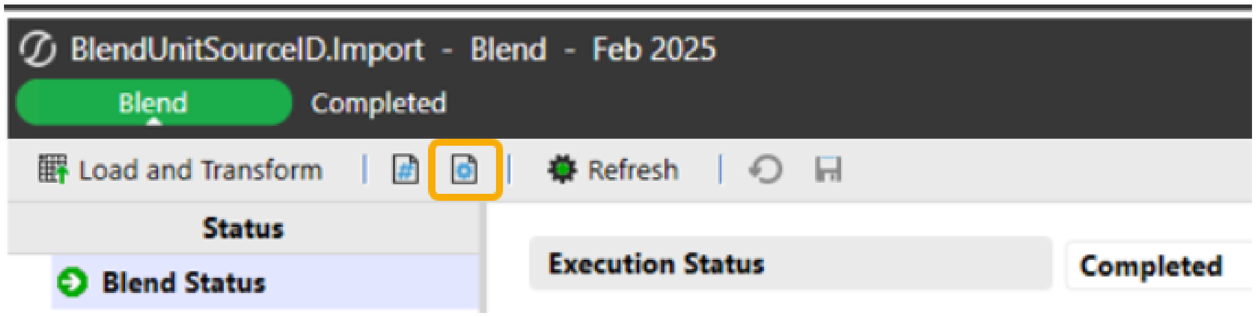
This status displays the state of the current table to denote the last time the table was updated and its size.



Execution Status	**Completed**
Table Name	BIB_GoldAnalyticBlend_BlendUnitSourceIDImport_Blend_2025M2
User Name	Administrator
Execution Time	1/10/2025 3:50:20 PM (L), 1/10/2025 3:50:20 PM (UTC)
Execution Row Count	15
Table Status	**Live (Last Execution SUCCESSFUL)**
Live Time	1/10/2025 3:50:20 PM (L), 1/10/2025 3:50:20 PM (UTC)
Live Row Count	15

BI Blend Processing Logs

Statistics and information are written to the log file upon successful completion of a Blend task.



BlendUnitSourceID.Import - Blend - Feb 2025

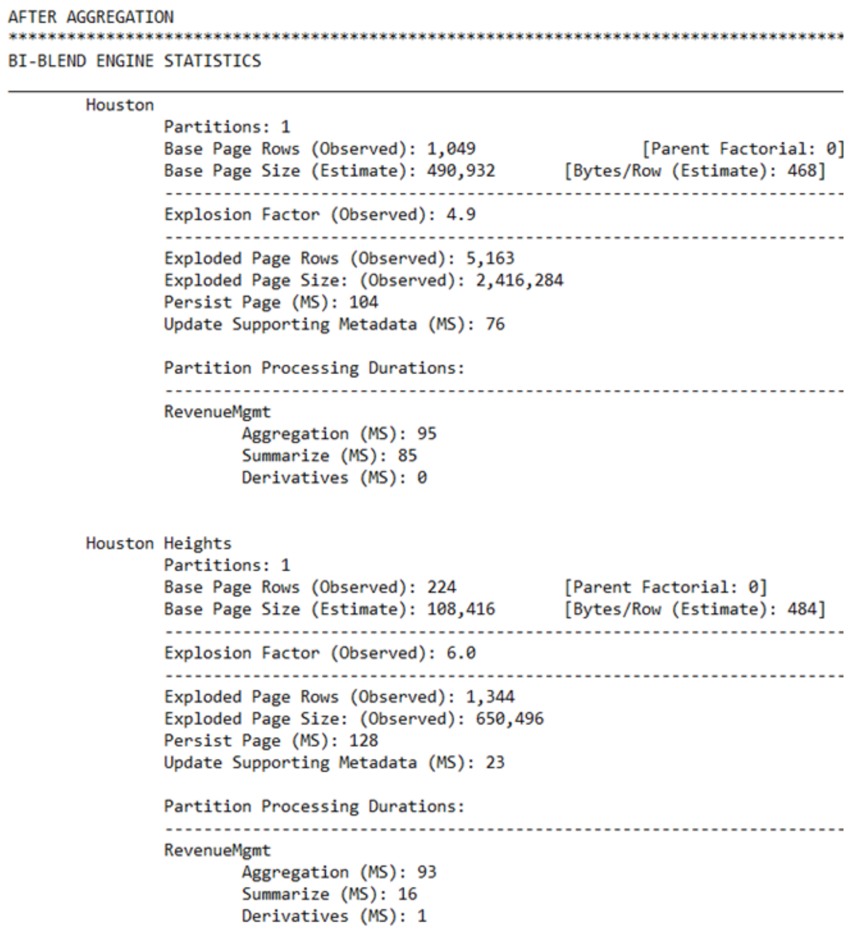
Blend Completed

Execution Status	**Completed**

Basic Log File Parsing

The log file provides statistics to manage the Blend task. The source file can identify data records. BI Blend generates records not only for the base member, but also on the parent level hierarchies defined by the aggregation points set using the workflow BI-Blend Settings. The Log File gives insight into this structure to estimate the processing needed on the current and potential future data being processed.

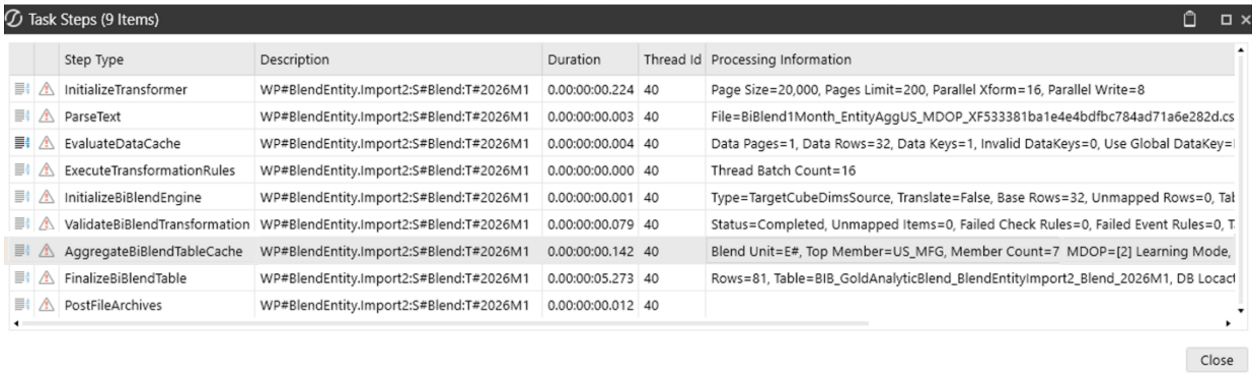
- **Blend Unit**: The identified dimension used as a partition, generating page records.
- **Base Page Rows**: The total number of unique base member records.
- **Exploded Page Rows**: The total number of rows within the page used to store the base member records and the hierarchical records.
- **Explosion Factor**: The number of data record cells generated by the base rows across the dimensions and parent levels. The factor is calculated by taking the number of Exploded Page Rows divided by the number of Total Base Page Rows. In the following example, Exploded Page Rows 1,344 is divided by Base Page Rows 224, which equals the Explosion Factor of 6.0.



```
AFTER AGGREGATION
************************************************************************************
BI-BLEND ENGINE STATISTICS
____________________________________________________________________________________
        Houston
                Partitions: 1
                Base Page Rows (Observed): 1,049                [Parent Factorial: 0]
                Base Page Size (Estimate): 490,932      [Bytes/Row (Estimate): 468]
                --------------------------------------------------------------------
                Explosion Factor (Observed): 4.9
                --------------------------------------------------------------------
                Exploded Page Rows (Observed): 5,163
                Exploded Page Size: (Observed): 2,416,284
                Persist Page (MS): 104
                Update Supporting Metadata (MS): 76

                Partition Processing Durations:
                --------------------------------------------------------------------
                RevenueMgmt
                        Aggregation (MS): 95
                        Summarize (MS): 85
                        Derivatives (MS): 0


        Houston Heights
                Partitions: 1
                Base Page Rows (Observed): 224          [Parent Factorial: 0]
                Base Page Size (Estimate): 108,416      [Bytes/Row (Estimate): 484]
                --------------------------------------------------------------------
                Explosion Factor (Observed): 6.0
                --------------------------------------------------------------------
                Exploded Page Rows (Observed): 1,344
                Exploded Page Size: (Observed): 650,496
                Persist Page (MS): 128
                Update Supporting Metadata (MS): 23

                Partition Processing Durations:
                --------------------------------------------------------------------
                RevenueMgmt
                        Aggregation (MS): 93
                        Summarize (MS): 16
                        Derivatives (MS): 1
```

Task Activity

The BI Blend Load and Transform Task Activity status monitors BI Blend processes. This tracking breaks out the various processing steps for analysis.



Step Type	Description	Duration	Thread Id	Processing Information
InitializeTransformer	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.224	40	Page Size=20,000, Pages Limit=200, Parallel Xform=16, Parallel Write=8
ParseText	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.003	40	File=BiBlend1Month_EntityAggUS_MDOP_XF533381ba1e4e4bdfbc784ad71a6e282d.cs
EvaluateDataCache	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.004	40	Data Pages=1, Data Rows=32, Data Keys=1, Invalid DataKeys=0, Use Global DataKey=
ExecuteTransformationRules	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.000	40	Thread Batch Count=16
InitializeBiBlendEngine	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.001	40	Type=TargetCubeDimsSource, Translate=False, Base Rows=32, Unmapped Rows=0, Tal
ValidateBiBlendTransformation	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.079	40	Status=Completed, Unmapped Items=0, Failed Check Rules=0, Failed Event Rules=0, T
AggregateBiBlendTableCache	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.142	40	Blend Unit=E#, Top Member=US_MFG, Member Count=7 MDOP=[2] Learning Mode,
FinalizeBiBlendTable	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:05.273	40	Rows=81, Table=BIB_GoldAnalyticBlend_BlendEntityImport2_Blend_2026M1, DB Locati
PostFileArchives	WP#BlendEntity.Import2:S#Blend:T#2026M1	0.00:00:00.012	40	

External Database Connection

The BI Blend target table must be a separate external database that is not in the Application Database. Update the Application Server Configuration Utility to confirm a connection to the new BI Blend target database.

NOTE: This is only for On-Premise customers. If SaaS customers need a separate external database created and a connection to that database, contact OneStream Support.

Calculations

Enrichment of the data processed by the BI Blend engine can be performed through the in-memory processing of records, or at run-time using Workspace Data Adapters. The use of existing metadata hierarchies is an optimal solution to add parent level aggregations and reporting points to the data set. Derivative Rules enrich the data set with time-based and other calculations. Run-time calculations can be designed into Dashboard SQL Data Adapters. The inclusion of aggregations and calculations adds to the record count of BI Blend processed data sets.

BI Blend Derivative Rules

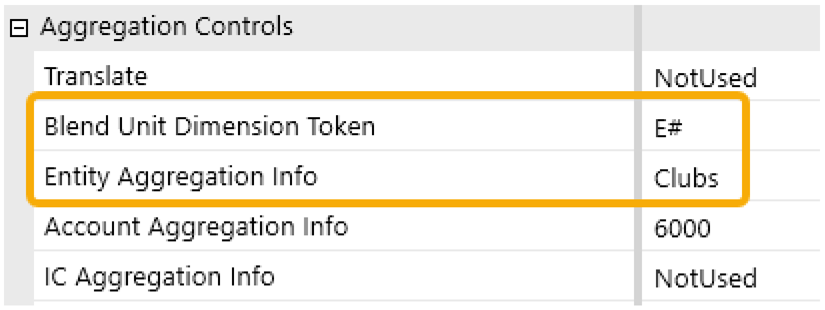
A special category of Derivative Rules is designed for the BI Blend Engine. The calculation of each derivative rule is performed within the Blend Unit. The calculations within the Blend Unit partition cannot reference data found in other Blend Unit partitions. If required, the summarization will be performed in the database or reporting layer.

The dependency of calculations on the Blend Unit is important to consider when designing a BI Blend Workflow. There are three classes of derivative rules specifically designed for the BI Blend Engine:

- **BlendUnit All**: Rules will process on all Blend Units.
- **BlendUnit Base**: Rules will process only on hierarchy base-level Blend Units.
- **BlendUnit Parent**: Rules will process only on parent-level Blend Units.

The use of each class of derivative rules depends on the design of the Blend Unit in the Workflow BI-Blend settings. First, determine which dimension is used to derive the Blend Unit by the Blend Unit Dimension Token selection. Second, understand and review the aggregation point selected for the Blend Unit Dimension Token.

Parent level members create base and parent level partitions. If a base-level member is selected, only a single partitioned Blend Unit page is created, and BlendUnit All is used to define the derivative rules.



Aggregation Controls	
Translate	NotUsed
Blend Unit Dimension Token	E#
Entity Aggregation Info	Clubs
Account Aggregation Info	6000
IC Aggregation Info	NotUsed

Derivative Rule Syntax

Record selection, data sub-setting, the renaming of members, and the creation of buffers for calculations on record sets requires special rule expression syntax.

Activity	Syntax and Example	Where Used
Record Selection	<Dimension token>#[<Member or wildcard>]=<Derived member> A#[Revenue]=Average	Rule expression

Activity	Syntax and Example	Where Used
Details: This syntax identifies the records in the import dataset on which to operate and sets the new member for the derived records. The new member does not need to be defined in the Dimension Library.		
Setting dimensions to a common member	:<Dimension token>#[*]=<Common member name> A#[Revenue]=Average:A1#[*]=None:A2 [*]=None	Rule expression
Details: Only records which are common are evaluated together in a calculation. This syntax can be used to set dimensions, other than DateTime dimensions, to a common member. The common member does not need to be defined in the Dimension Library.		
xBDT expression to perform time-based grouping of records	xBDT:[<Start DateTime>~<End DateTime>]!GroupingLevel {Y, HY, Q, M, W, D ,TH, TM, TS, All, Unknown} A#[Revenue]=Average:A1#[*]=None:A2# [*]=None:xBDT#[2025-01-01 00;00;01~2025-12-31 23;59;59]!D	Rule expression

Activity	Syntax and Example	Where Used
Details: This syntax subsets records into groups based on their timestamps. The time unit options range from year (Y) to seconds (TS). Other options are Unknown, which groups based on the level of precision of the timestamps, and All, which groups based on the entire range. All is useful for grouping across multiple years. The datetime format for this expression is: YYYY-MM-DD hh;mm;ss but can be less precise, for example as YYYY-MM-DD.		
Special xBDT time references	CurrentDateTimeLocal {Y, HY, Q, M, W, D, TH, TM, TS}+/-X A#[Revenue]=Average:A1#[*]=None:A2#[*]=None:xBDT#[D-7~CurrentDateTimeLocal]!D	Rule expression
Details: DateTime references can be used instead of explicitly setting the start or end date in an xBDT range. This returns the current DateTime of the system server and counts forward or backwards from the range limit by an integer (X) in the unit of the time unit specified		
Creating a buffer 1	--<Dimension token>#[<member name>]=<temporary member name> --A#[Expense]=Temp_Buf	Math value field of the Logical operator

Activity	Syntax and Example	Where Used
Details: Buffers can be used in place of an operand for some Logical operations. The syntax creates a buffer, places records for the indicated member in it, and assigns a temporary member name for the purposes of calculation.		
Creating a buffer 2	--<Buffer name>=<Dimension token>#[<member name>]=<temporary member name>	Rule expression
Details: Buffers can be created in the Rule expression for use with Complex Expressions and Business Rules. The syntax sets the buffer name, places records for the indicated member in it, and assigns a temporary member name for the purposes of calculation		
Selecting the DateTime fields for Col DateDiff operators	//<DateTime dimension token>; <DateTime dimension token> //xDt1;xDt2	Math value field of the Logical operator
Details: The Col DateDiff suite of operators calculate the difference between two timestamps in the same record. The syntax configures the DateTime dimensions to use in the calculation.		

Time-Aware Derivative Logical Operators

Blend X Derivative statistical computations filter records by time filters. This enables you to develop dynamic standardized calculations to group records by time that function across all data sets. Time filtering is keyed off the primary date field xBlendDateTime, as xBDT, for data series statistical calculations.

Date Grouping

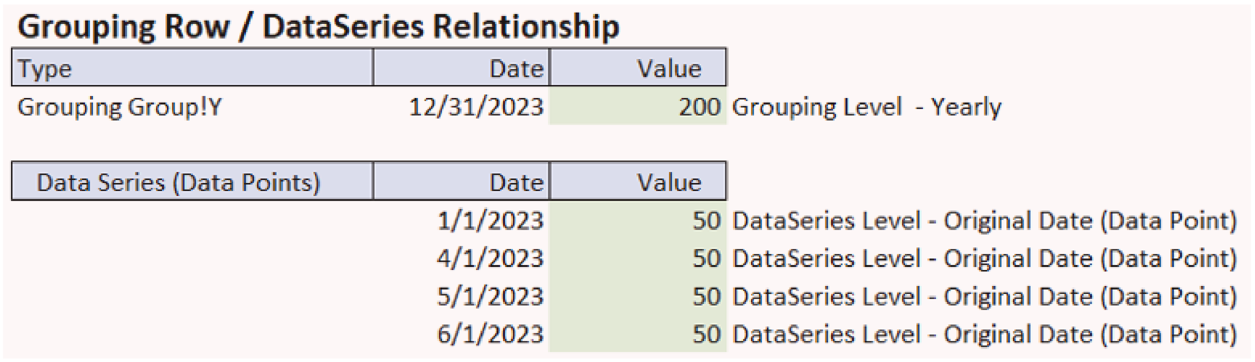
Date groupings are appended to the Blend X time-aware functions to dynamically filter the records by a date range. The criterion for the range includes the start and end range, as well as the grouping level.

Expression Syntax for Statistical Grouping
xBDT:[<Start DateTime>~<End DateTime>]!GroupingLevel {Y, HY, Q, M, W, D ,TH, TM, TS}

Grouping Expression Example
xBDT:[YYYY-MM-DD hh;mm;ss~YYYY-MM-DD hh;mm;ss]!Y

Example
A#[TicketCost]=Stats_SumCost:A1#[*]=None:A2#[*]=None:A4#[*]=None:xBDT#[2022-01-01 00;00;00~2022-12-31 23;59;59]!Y

All records within the Blend Unit are filtered by the range. Results are generated at the level of detail defined in the Grouping Level. As an example, grouping a full year data set by Y will return a single result for each year in the data set. Similarly, specifying Q will return four quarterly records.



Grouping Row / DataSeries Relationship

Type	Date	Value	
Grouping Group!Y	12/31/2023	200	Grouping Level - Yearly

Data Series (Data Points)	Date	Value	
	1/1/2023	50	DataSeries Level - Original Date (Data Point)
	4/1/2023	50	DataSeries Level - Original Date (Data Point)
	5/1/2023	50	DataSeries Level - Original Date (Data Point)
	6/1/2023	50	DataSeries Level - Original Date (Data Point)

Rolling Time-Aware Filters

Rolling time-aware filters are appended to the Blend X time-aware functions to dynamically calculate rolling calculations. All records within the Blend Unit are filtered by range. The syntax references the date, operation for the range, and grouping level. A fixed date or variable is supported.

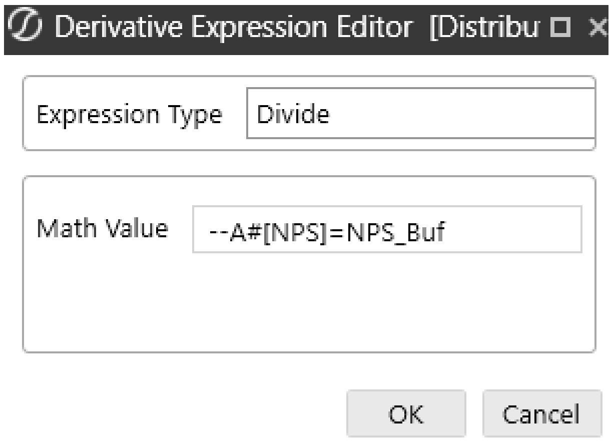
Expression Syntax for Rolling Date
xBDT:[<D-X>~<Ref DateTime>]!GroupingLevel {Y, HY, Q, M, W, D ,TH, TM, TS}

Rolling Date Example
xBDT:D-90~YYYY-MM-DD hh;mm;ss]!Y

Example
A#[TicketCost]<<MyPrefix_:A1#[*]=None:A2#[*]=None:A4#[*]=None:xBDT#
[CurrentDateTimeLocal~D+6]!Y

Data Buffer Logic

The Blend X logical operators support data buffer math. Calculation operations are not restricted to defined constants. The secondary buffer is applied to the **Math Value** field in the complex expression.


Derivative Expression Editor [Distribu

Expression Type: Divide

Math Value: --A#[NPS]=NPS_Buf

Blend X Derivative Naming Convention

BI Blend table results maintain account type references for data records. When enriching the data set using derivative rules, the rule naming convention property allows you to apply Account Type to each of the data records generated. Account Type prefixes are as follows:

- **R**: Revenue
- **E**: Expense
- **N**: Non-Financial
- **A**: Asset
- **L**: Liability

Logical Operator Definitions

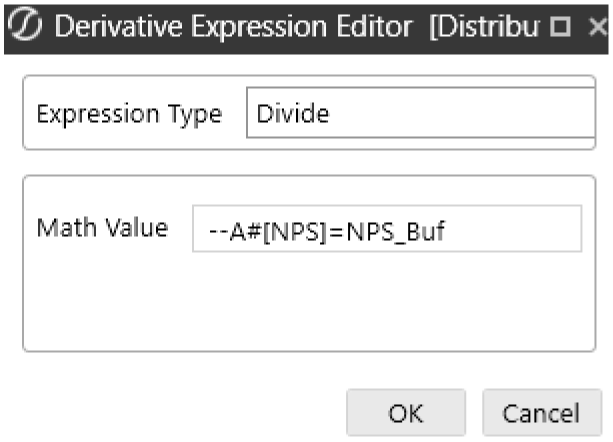
The section provides information about logical operators.

Add
Adds the derivative member’s value by the value set in the Math Value field.

Subtract
Subtracts the value specified in the Math Value field from the derivative member’s value.

Multiply
Multiplies the derivative member’s value by the value specified in the Math Value field.

Divide
Divides the derivative member’s value by the value specified in Math Value. You can also use a buffer value. For example, --A#[NPS]=NPS_Buf.

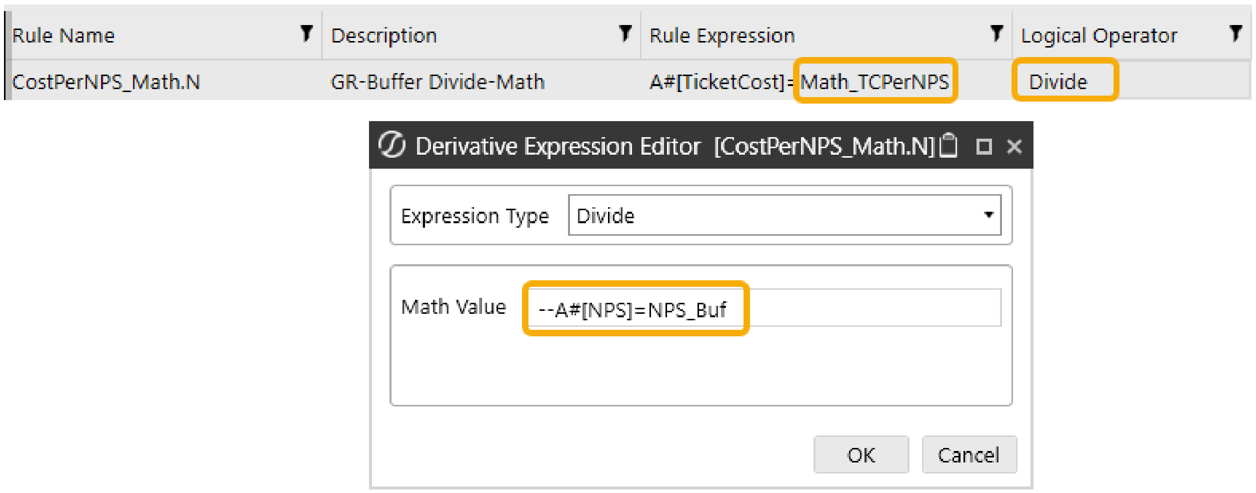


Derivative Expression Editor [Distribu

Expression Type: Divide

Math Value: --A#[NPS]=NPS_Buf

In this example, accounts named TicketCost in the load file are targeted. These accounts are renamed to Math_TCPerNPS.



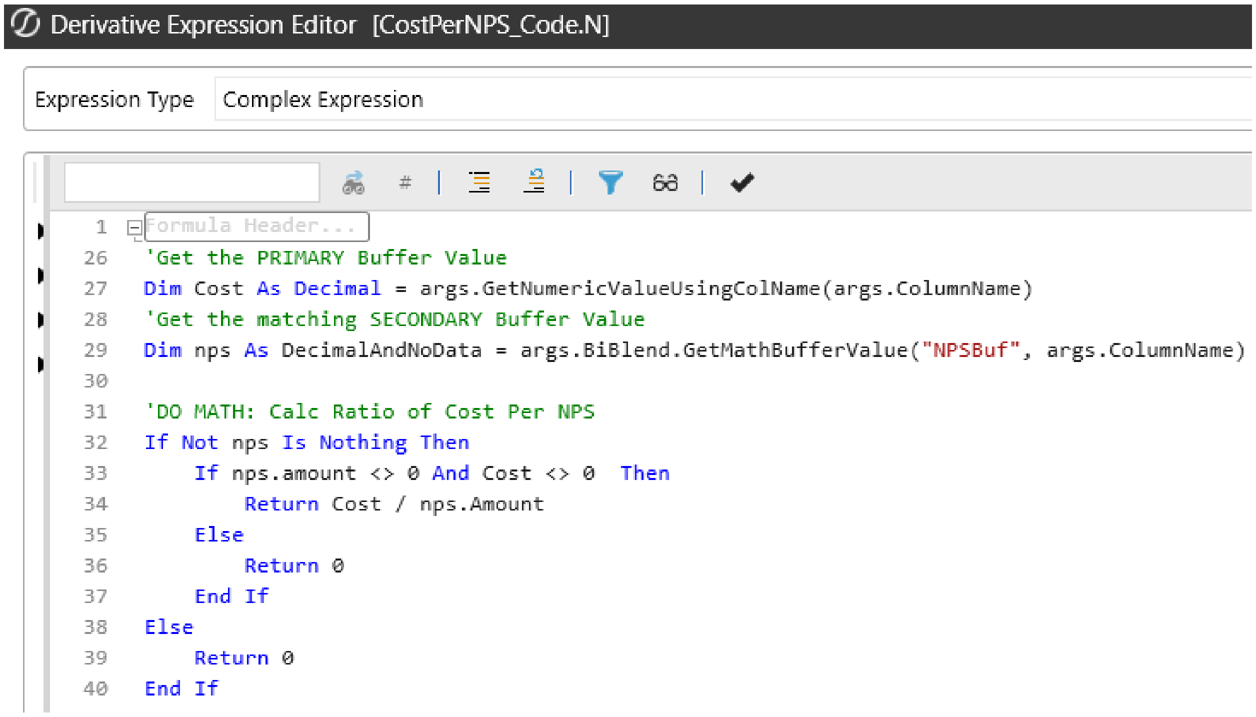
Rule Name	Description	Rule Expression	Logical Operator
CostPerNPS_Math.N	GR-Buffer Divide-Math	A#[TicketCost]=Math_TCPerNPS	Divide

Derivative Expression Editor [CostPerNPS_Math.N]

Expression Type: Divide

Math Value: --A#[NPS]=NPS_Buf

This is the complex expression that performs the division. The MathBufferValue amount is captured for the NPS account. The values for the original TicketCost account records are then divided by the value of the NPS account.

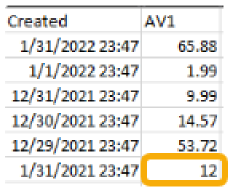

Derivative Expression Editor [CostPerNPS_Code.N]

Expression Type: Complex Expression

```
Formula Header...
'Get the PRIMARY Buffer Value
Dim Cost As Decimal = args.GetNumericValueUsingColName(args.ColumnName)
'Get the matching SECONDARY Buffer Value
Dim nps As DecimalAndNoData = args.BiBlend.GetMathBufferValue("NPSBuf", args.ColumnName)

'DO MATH: Calc Ratio of Cost Per NPS
If Not nps Is Nothing Then
    If nps.amount <> 0 And Cost <> 0  Then
        Return Cost / nps.Amount
    Else
        Return 0
    End If
Else
    Return 0
End If
```

Lag (Years)

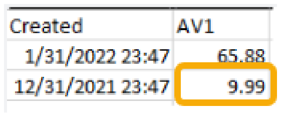
Returns the past value (lag) per the number of years set in the Math Value field of the rule. The Math Value equals one by default. The lag operator looks at the first date in the set, then returns the lag value for the number of years specified. In the data set shown in the example, the value 12 would be returned if the Lag (Years) value is set to one.



Created	AV1
1/31/2022 23:47	65.88
1/1/2022 23:47	1.99
12/31/2021 23:47	9.99
12/30/2021 23:47	14.57
12/29/2021 23:47	53.72
1/31/2021 23:47	12

Lag (Months)

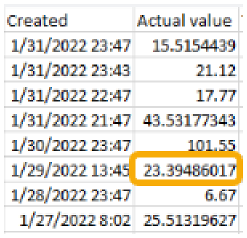
Returns the past value (lag) per the number of months set in the Math Value field of the rule. The Math Value equals one by default. The lag operator looks at the first date in the set, then returns the lag value for the number of months specified. In the data set shown below, the value 9.99 would be returned if the Lag (Months) value is set to one.



Created	AV1
1/31/2022 23:47	65.88
12/31/2021 23:47	9.99

Lag (Days)

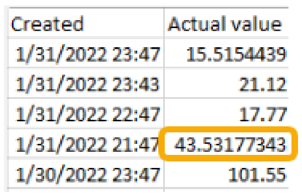
The Lag (Days) logical operator returns the past value (lag) per the number of days set in the Math Value field of the rule. The data set is first sorted in descending order by the Created date. The lag operator looks at the first date in the set, then returns the lag value for the number of days specified. In the data set shown in the example, the value 23.39486017 would be returned if the Lag (Days) value is set to two.



Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77
1/31/2022 21:47	43.53177343
1/30/2022 23:47	101.55
1/29/2022 13:45	23.39486017
1/28/2022 23:47	6.67
1/27/2022 8:02	25.51319627

Lag (Hours)

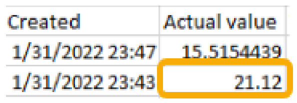
Returns the past value (lag) per the number of hours set in the Math Value field of the rule. The data set is first sorted in descending order by the Created date/time. The lag operator looks at the first date/time in the set, then returns the lag value for the number of hours specified. In the data set shown in the example, the value 43.53177343 would be returned if the Lag (Hours) value is set to two.



Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77
1/31/2022 21:47	43.53177343
1/30/2022 23:47	101.55

Lag (Minutes)

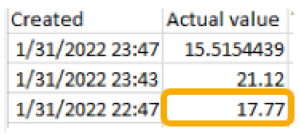
Returns the past value (lag) per the number of minutes set in the Math Value field of the rule. The data set is first sorted in descending order by the Created date/time. The lag operator looks at the first date/time in the set, then returns the lag value for the number of minutes specified. In the data set shown in the example, the value 21.12 would be returned if the Lag (Minutes) value is set to four.



Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12

Lag (Seconds)

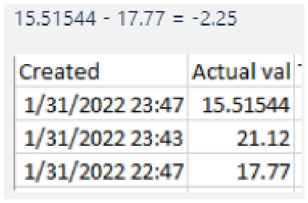
Returns the past value (lag) per the number of seconds set in the Math Value field of the rule. The data set is first sorted in descending order by the Created date/time. The lag operator looks at the first date/time in the set, then returns the lag value for the number of seconds specified. In the data set shown in the example, the value 17.77 would be returned if the Lag (Seconds) value is set to 3600 (60 minutes).



Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77

Lag Change (Seconds)

Returns the difference between the latest value and a past value (lagged value), per the number of seconds set in the Math Value field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of seconds set in the rule. The difference between the latest value and the lag value is returned. In the data set shown in the example, the value -2.25 would be returned if the Lag Change (Seconds) value is set to 3600 (60 minutes).

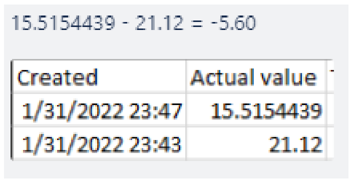


15.51544 - 17.77 = -2.25

Created	Actual val
1/31/2022 23:47	15.51544
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77

Lag Change (Minutes)

Returns the difference between the latest value and a past value (lagged value), per the number of minutes set in the Math Value field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of minutes set in the rule. The difference between the latest value and the lag value is returned. In the data set shown in the example, the value -5.60 would be returned if the Lag Change (Minutes) value is set to four.

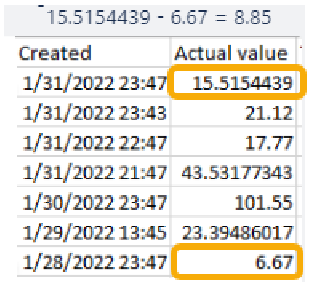


15.5154439 - 21.12 = -5.60

Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12

Lag Change (Days)

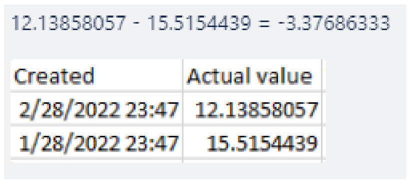
Returns the difference between the latest value and a past value (lagged value), per the number of days set in the Math Value field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of days set in the rule. The difference between the latest value and the lag value is returned. In the data set shown in the example, the value 8.85 would be returned if the Lag Change (Days) value is set to three.


15.5154439 - 6.67 = 8.85

Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77
1/31/2022 21:47	43.53177343
1/30/2022 23:47	101.55
1/29/2022 13:45	23.39486017
1/28/2022 23:47	6.67

Lag Change (Months)

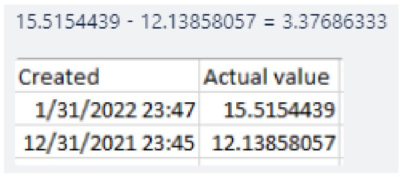
Returns the difference between the latest value and a past value (lagged value), per the number of months set in the Math Value field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of months set in the rule. The difference between the latest value and the lag value is returned. In the data set shown in the example, the value -3.37686333 would be returned if the Lag Change (Months) value is set to one.


12.13858057 - 15.5154439 = -3.37686333

Created	Actual value
2/28/2022 23:47	12.13858057
1/28/2022 23:47	15.5154439

Lag Change (Years)

Returns the difference between the latest value and a past value (lagged value), per the number of years set in the Math Value field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of years set in the rule. The difference between the latest value and the lag value is returned. In the data set shown in the example, the value 3.37686333 would be returned if the Lag Change (Years) value is set to one.

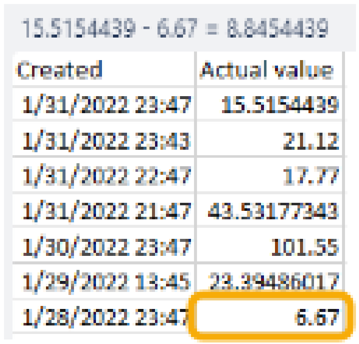


15.5154439 - 12.13858057 = 3.37686333

Created	Actual value
1/31/2022 23:47	15.5154439
12/31/2021 23:45	12.13858057

Lag Change (Step Back)

First, sorts a data set in descending order by the Created date. Then, it Steps Back from the first record in the data set and selects the record based on the number of positions back specified in the Math Value field. The first record in the data set is position zero. In the data set in the example, the difference between the 1st record and the 6th record is returned.



15.5154439 - 6.67 = 8.8454439

Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77
1/31/2022 21:47	43.53177343
1/30/2022 23:47	101.55
1/29/2022 13:45	23.39486017
1/28/2022 23:47	6.67

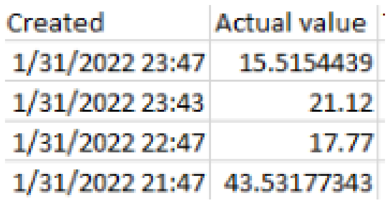
Lag (Step Back)

Sorts a data set in descending order by the Created date and steps back from the first record in the data set, selecting the record based on the number of positions back specified in the Math Value field.

Lag Change (Hours)

Returns the difference between the latest value and a past value (lagged value), per the number of hours set in the Math Value field. The data set is first sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of hours. The difference between the latest value and the lag value is returned.

In the data set shown in the example, the value -28.02 would be returned if the Lag Change (Hours) Math Value is set to two, where 15.5154439 - 43.53177343 = -28.02.



Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77
1/31/2022 21:47	43.53177343

Absolute Value

Generates a new record that contains the absolute value of the values in the load file.

Round (Precision)

Generates a new record that contains the rounded value of the values in the load file. The Math.Round(decimal, int) method is used. The Math Value is the number of decimal places in the return value. It rounds a decimal value to the specified number of decimal places and rounds midpoint values to the nearest even number.

Modulo (Divisor)

Calculates the modulo (remainder after division) value. For example, 12/10 modulo = 2. The divisor is set in the Math Value field within the Derivative Expression Editor dialog box. The default divisor value is 10.

Power (Exponent)

Generates a new record that contains the exponential value of the values in the load file. For example, five to the power of two is 5 x 5 = 25. The power value is set in the Math Value field of the Derivative Expression Editor dialog box.

Proportion (Count)

Calculates the proportion of one out of the total number of rows that were created.

> **Example:** In the rule expression A#[TicketCost]=Stats_PropCount:A1#[]=None:A2#[]=None:xBDT#[2022-01-01~2022-12-31]!Y, a derivative record is created for each record where the account equals TicketCost and that falls within the date range 2022-01-01 - 2022-12-31. If 54 total rows are created, the proportion count value will equal 1/54 = 0.018518519.

Proportion (Value)

Calculates the proportion of the value of a field out of the total for that column, where Value / Sum of column = Proportion (Value).

> **Example:** In the rule expression A#[TicketCost]=Stats_PropValue:A1#[]=None:A2#[]=None:xBDT#[2022-01-01~2022-12-31]!Y, a derivative record is created for each record where the account equals TicketCost, and that falls within the date range 2022-01-01 to 2022-12-31.

Clip (Lower Bound)

Sets a minimum lower threshold for records that are created as a result of this rule. The default value is 10. Values less than the lower bound are set to the lower bound value. For example, if the original value is nine, the value is set to 10 in the derived record.

Clip (Upper Bound)

Sets an upper threshold for records that are created as a result of this rule. The default value is 35. Values greater than the upper bound are set to the upper bound value. For example, If the original value is 36, the value is set to 35 in the derived record.

Count

Creates one record in the BI Blend table. This is a count of the total number of rows returned per the rule expression.

First DateTime

Creates one record in the BI Blend table that shows the value for the earliest created date/time of the records matching the rule expression.

Last DateTime

Creates one record in the BI Blend table that shows the value for the latest date/time of the records that match the rule expression.

Minimum Value

Creates one record in the BI Blend table that shows the lowest value for the records that match the rule expression.

Maximum Value

Creates one record in the BI Blend table that shows the highest value for the records that match the rule expression.

Average

Creates one record in the BI Blend table that shows the average value for the records that match the rule expression.

Median

Creates one record in the BI Blend table that shows the median value for the records that match the rule expression.

Mode

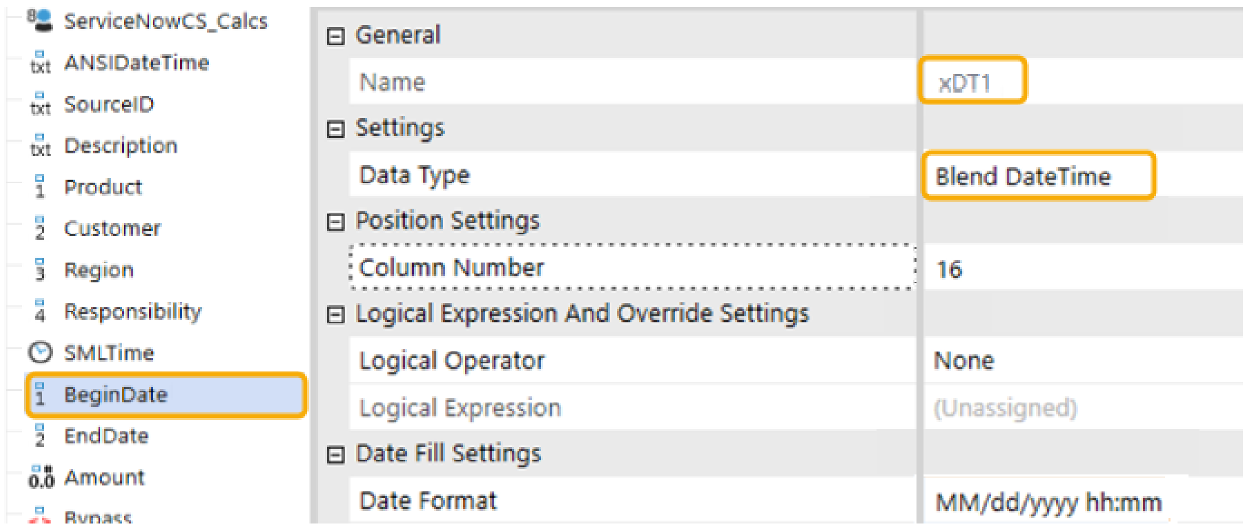
Creates one record in the BI Blend table that shows the mode value for the records that match the rule expression. The mode is the value that appears the most number of times in a data set. If there are multiple instances of a mode in a file, the first value is returned if the values are sorted ascending. This is also done for the mode calculation in Excel.

Standard Deviation

Creates one record in the BI Blend table that shows the standard deviation value for the records that match the rule expression. It uses the standard deviation based on the entire population. Numbers can be validated by using the STDEV.P function in Excel.

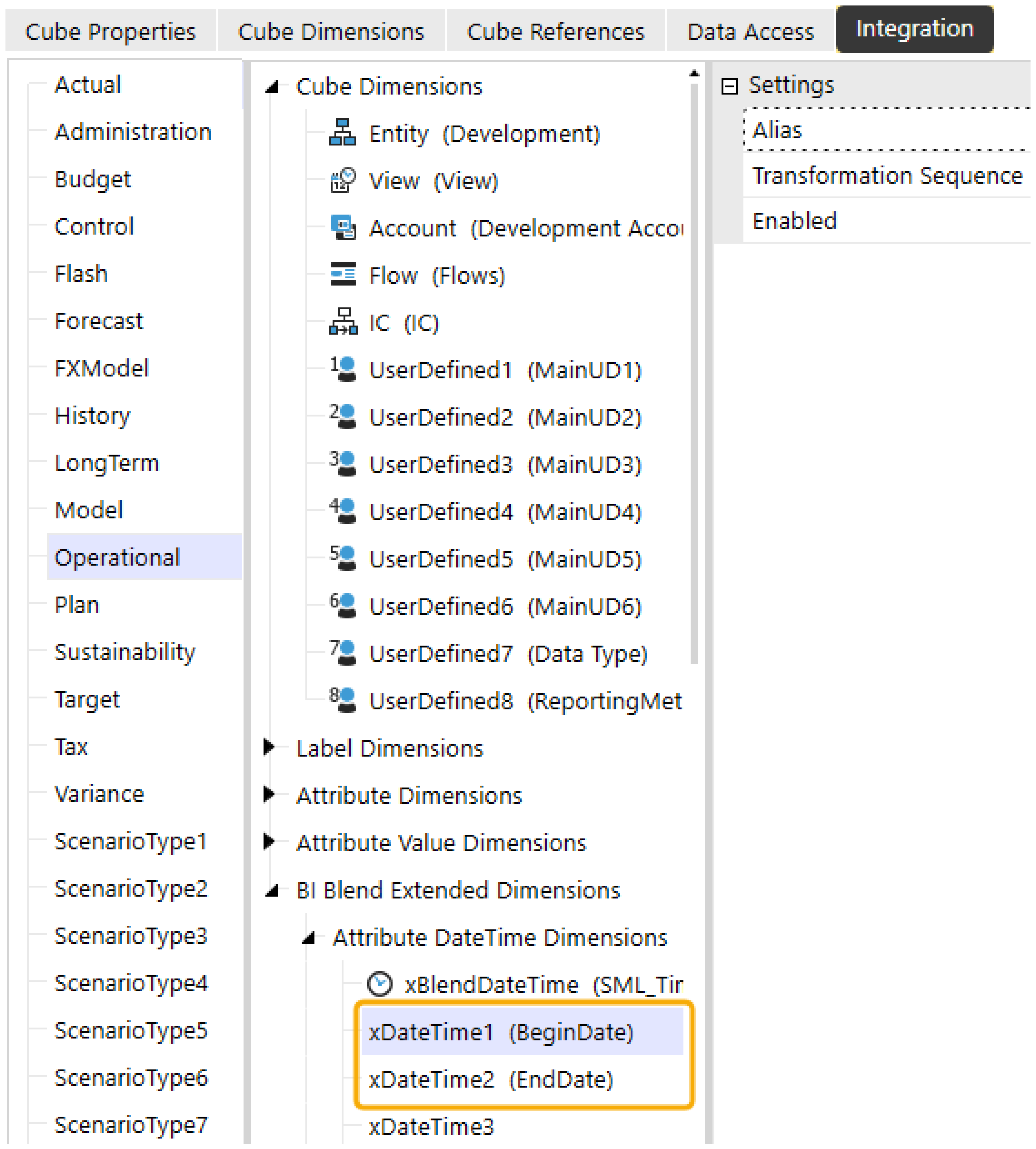
Col DateDiff (Days)

Returns the number of days within the BeginDate and EndDate specified in the load file. The value is rounded down to the nearest whole number. The Math Value syntax, //xDt1;xDt2, indicates that the difference between EndDate and BeginDate is returned. XDt1 and xDt2 are aliases for BeginDate and EndDate, which are set in the Data Source.



General	
Name	xDT1
Settings	
Data Type	Blend DateTime
Position Settings	
Column Number	16
Logical Expression And Override Settings	
Logical Operator	None
Logical Expression	(Unassigned)
Date Fill Settings	
Date Format	MM/dd/yyyy hh:mm

BI Blend DateTime dimensions are set on the cube.



Here is an example of a rule expression for this operator: A#[TicketCost]=Stats_ DateDiffDays1and2:A1#[]=None:A2#[]=None:xBDT#[2022-01-01 00;00;00~2022-12-31 23;59;59]!D. This filters the data series to return account data between the specified date range and sets the Group Level by day.

For all accounts named TicketCost in the load file, create a new record and complete the following steps:

1. **DateDiffDays.E**: the .E portion of this rule name sets the Account Type to Expense.
2. Change the account name to **Stats_DateDiffDays1and2**.
3. Set all A1 instances (Product column in db, Instance column from load file) to **None**.
4. Set all A2 instances (Customer column in db, Instance col from load file) to **None**.

Col DateDiff (Months)
Returns the number of months within the BeginDate and EndDate specified in a load file. The value is rounded down to the nearest whole number. The Math Value syntax, //xDt1;xDt2, indicates that the difference between EndDate and BeginDate is returned. XDt1 and xDt2 are aliases for BeginDate and EndDate, which are set in the Data Source.

> **Example:** A#[TicketCost]=Stats_DateDiffMonths:A1#[]=None:A2#[]=None:xBDT#[2022-01-01 00;00;00~2022-12-31 23;59;59]!M. :xBDT#[2022-01-01 00;00;00~2022-12-31 23;59;59] filters the data series to return account data in the date range specified. !M sets the Group Level by Month.

Col DateDiff (Years)
Returns the number of years within the BeginDate and EndDate specified in a load file. The value is rounded down to the nearest whole number. The Math Value syntax, //xDt1;xDt2, indicates that the difference between EndDate and BeginDate is returned. XDt1 and xDt2 are aliases for BeginDate and EndDate, which are set in the Data Source.

Col DateDiff (Hours)
Returns the number of hours within the BeginDate and EndDate specified in a load file. The value is rounded down to the nearest whole number hour. The Math Value syntax, //xDt1;xDt2, indicates that the difference between EndDate and BeginDate is returned. XDt1 and xDt2 are aliases for BeginDate and EndDate, which are set in the Data Source.

Col DateDiff (Minutes)
Returns the number of minutes within the BeginDate and EndDate specified in a load file. The value is rounded down to the nearest whole number minute. xDt1 and xDt2 are aliases for BeginDate and EndDate, which are set in the Data Source.

Col DateDiff (Seconds)
Returns the number of seconds within the BeginDate and EndDate specified in a load file. xDt1 and xDt2 are aliases for BeginDate and EndDate, which are set in the Data Source.

Log (P+1)
Generates a new record that contains the natural log (1+x) value of the values in the load file. The following function is used in the calculation: Math.Log() function. Natural Log (1+x).

> **NOTE:** The formula reference in BI Blend for the Log P1 function is not calculated for values <=1 and returns null in that case.

Apply Suffix or Prefix
Renames accounts to either MyPrefix_TicketCost or TicketCost_MySuffix using the following expressions: A#[TicketCost]>>MyPrefix_ or A#[TicketCost]<<_MySuffix, respectively.

Create If (> x)
Creates a record in the BI Blend table for each record where the Actual Value is greater than the value set in the Math Value field.

Create If (< x)
Creates a record in the BI Blend table for each record where the Actual Value is less than the value set in the Math Value field.

Filtering Records with Rule Expressions

Rule expressions in transformation rules can be used to filter records by date and time ranges using xBDT syntax. When data is loaded, derivative records are created for records that fall within the specified date and time range. Here are some examples:

xBDT#[2022-01-01 00;00;00~2022-01-31 23;59;59]: Creates derivative records for imported records that are >= start date/time and <= finish date/time.

xBDT#[CurrentDateTimeLocal~D+6]: Creates derivative records for imported records that are greater than the current date/time plus 6 days.

xBDT#[D-90~2022-03-31 23;59;59]: Creates derivative records for imported records that fall within the range of the specified date/time minus 90 days. Records equal to the current date/time are imported as well.

Sub-String Grouping in Rule Expressions

Syntax used in a rule expression can capture a specified number of characters from a string, starting with the character in the first position. In the example below, the first six characters of an account name are captured and applied in derivative records. Loaded records with the account name TicketCost will show an account name of Ticket in the derived records: A#[TicketCost]@6

Syntax used in a rule expression can capture a sub-string between the first and last characters of a string. In the example below, a four character sub-string is captured starting at the 7th character of the string. Loaded records with the account name TicketCost will show an account name of Cost in the derived records: A#[TicketCost]@7,4.

Logical Operators Usage

Logical Operators extend a normal mapping rule with VB.Net scripting functionality. To use logical operators, complete the following steps:

1. Navigate to **Application** > **Data Collection**>**Transformation Rules**.
2. Under **Rule Groups**, expand **Derivative**, then select a derivative rule group.
3. Select a value from the **Type** drop-down menu.
4. Click **Insert Row**.
5. Type a name into the **Rule Name** field.
6. Type a description into the **Description** field.
7. Type an expression into the **Rule Expression** field. For example, A#[TicketCost]=Math_Power2:A1#[]=None:A2#[]=None:xBDT#[2022-01-01 00;00;00~2022-12-31 23;59;59]!Y.
8. Double-click the new row in the **Logical Operator** column.
9. In the **Derivative Expression Editor** dialog box, from the **Expression Type** drop-down menu, select the logical operator you want to use.
10. Type an appropriate value into the **Math Value** field and click the **OK** button.
11. In the **Derivative Type** column, set the value to **Final (Exclude Calc)**.
12. In the **Order** column, type a value that is greater than the current highest order value in the list.
13. Click the **Save** button.
14. Navigate to the **OnePlace** tab and set a BI Blend Workflow POV.
15. Click **Load and Transform**.
16. Select a file and click **Open**.
17. Click the **OK** button.
18. Log in to the database server for the environment and execute a query on the BI Blend database where the data was loaded to. You will see records returned.

19. Verify that the values in the applicable amount column are equal to the expected results.

Fast Data Extract BRApis

Fast Data Extract (FDX) BRApis enable you to extract data from a variety of sources. These BRApis can be written directly in a connector business rule, or the connector business rule can call a Workspace Assembly rule that contains the FDX BRApis. The connector rule is then referenced by the data source and used to import data into BI Blend. FDX BRApis can be combined with SQL scripts and filters to extract only the data needed.

There are different FDX BRApis available depending on the source you want to extract data from:

- **OneStream Application**: FDX BRApis can extract data from different places in your OneStream application.
    - **Cube Views**: Extract data through a Cube View definition. This is ideal for defining data definitions through a Cube View, including Dynamic Calc results.
    - **Data Unit Members**: Extract cube data to a BI Blend target table through defined Data Unit filters.
    - **Stage Workflow Imports**: The ability to leverage existing Stage data. Uses include reporting on existing attribute records contained in Stage or simply enhanced dashboard reporting on Stage data.
- **Warehouse Data Source**: FDX BRApis can extract data from internal or external warehouse data.

- **Internal**: This refers to tables within your application database.

  Example: You can extract all or a subset of data from Solution Exchange solution tables and import them into BI Blend. Derivative rules can be used to further transform or perform calculations on data during the import.

- **External**: This refers to external databases that are available to you.

  Example: You can use FDX APIs to extract a subset of data from your BI Blend tables and import them into BI Blend. As with internal tables, derivative rules can be used to transform and perform calculations on the data during import.

FDX Use and Benefits

Key differentiators between FDX BRApis and other collection methods are the support of parallel processing, in-memory processing, and management of the Time dimension.

Performance is enhanced because the BRApi is able to parallel process. For example, extracting data by cube Data Unit will parallel process all the Data Units defined in the filter. The FDX BRApis do not generate a .csv file as data management Export Data or Export File processes do. The results of the export are managed during the BI Blend in-memory processing.

In cases of very large datasets, where multiple periods are loaded, the processing time can be slow because each period is reflected as a data record. FDX BRApis offer solutions to pivot the Time records to columns to create a matrix data layout. See Time Pivot. The data source can associate each of the periods with an Attribute Value or xBlend Attribute Value dimension within the Integration settings. The design will treat each record as a collection of 36 periods when processing.

Review the following examples of how FDX BRApis can be used to move data:

- Can be used to extract OneStream data and metadata so that it can be used in external systems
- Can be used to pull data from Solution Exchange for use in dashboard reports
- Can be used to export dynamically calculated results from Cube Views
- Can be used to extract data from external systems into OneStream
- Can be used to pull transient data from external sources at a more frequent interval than monthly or weekly
- Can be used to create additional data points for use with SensibleAI Forecast

FDX BRApis

The FDX API functions are used within a Connection Action in a connector business rule or Workspace Assembly File. They are split into separate groups: Return and Extract.

Return

These API functions are used to return field column names to the Data Source Connector Fields.

The Connector Action Type GetFieldList returns the field column names to the Data Source Connector Fields using FDX APIs from the following data sources:

Data Source	FDX API
Cube View	FdxGetCubeViewOrDataUnitColumnList
Data Unit	
Stage	FdxGetStageTargetColumnList
Warehouse	FdxGetWarehouseColumnList

Extract

These API functions are used to extract data using Fast Data Extract.

The Connector Action Type GetData runs these FDX APIs to extract data so that it can be imported. Data can be extracted from the following data sources:

Data Sources	FDX API	Description
Cube View	FdxExecuteCubeView	Extracts data defined through a Cube View. Any data presented in the Cube View is extracted, such as Dynamic Calculated results.
	FdxExecuteCubeViewTimePivot	Cube View data generates all Time as columns, which can be assigned as Attribute Value members in the Data Source.

Data Sources	FDX API	Description
Data Unit	FdXExecuteDataUnit	A cube data extract solution to extract data from Data Unit members.
	FdXExecuteDataUnitTimePivot	A cube data extract solution to extract data from Data Unit members. It generates all Time as columns, which can be assigned as Attribute Value members in the Data Source.
Stage	FdxExecuteStageTargetTimePivot	To extract existing workflow Stage data. It is extracting data from your Stage Target columns. It generates all Time as columns, which can be assigned as Attribute Value members in the Data Source.
Warehouse	FdXExecuteWarehouseTimePivot	To extract data from an external source system.

NOTE: You may need to use a Parser Rule to ensure your timestamp data conforms to the format required by Blend: MM/dd/yyyy hh:mm:ss

Time Pivot

FDX APIs with the suffix TimePivot enable you to extract multiple time periods and treat Time as a measure. To import multiple time periods as measures, enable the Attribute Value and xBlend Attribute Value dimensions, depending on how many time periods are needed. Once enabled, add them to the data source and map them to the source data fields.

Treating time as a measure improves performance when working with multiple time periods. Using FDX APIs results in flatter in-memory tables. If you pivot Time in a data table containing multiple time periods, the data for each time period is stored in separate columns. For example, the following tables display the same amount of data. One table stores Time as a dimension. The other table has pivoted Time and stores it as a measure. The table where time is stored as a measure is flatter and is faster to process due to fewer rows.

Time as a Dimension
The time periods are in the Time column, and there is a separate Amount column to store the values.

Entity	Scenario	Time	Account	Amount
E1	Actual	2025M1	Account Receivable	150
E1	Actual	2025M1	Gross Revenue	150
E1	Actual	2025M1	Cash	50
E1	Actual	2025M2	Account Receivable	250
E1	Actual	2025M2	Gross Revenue	250

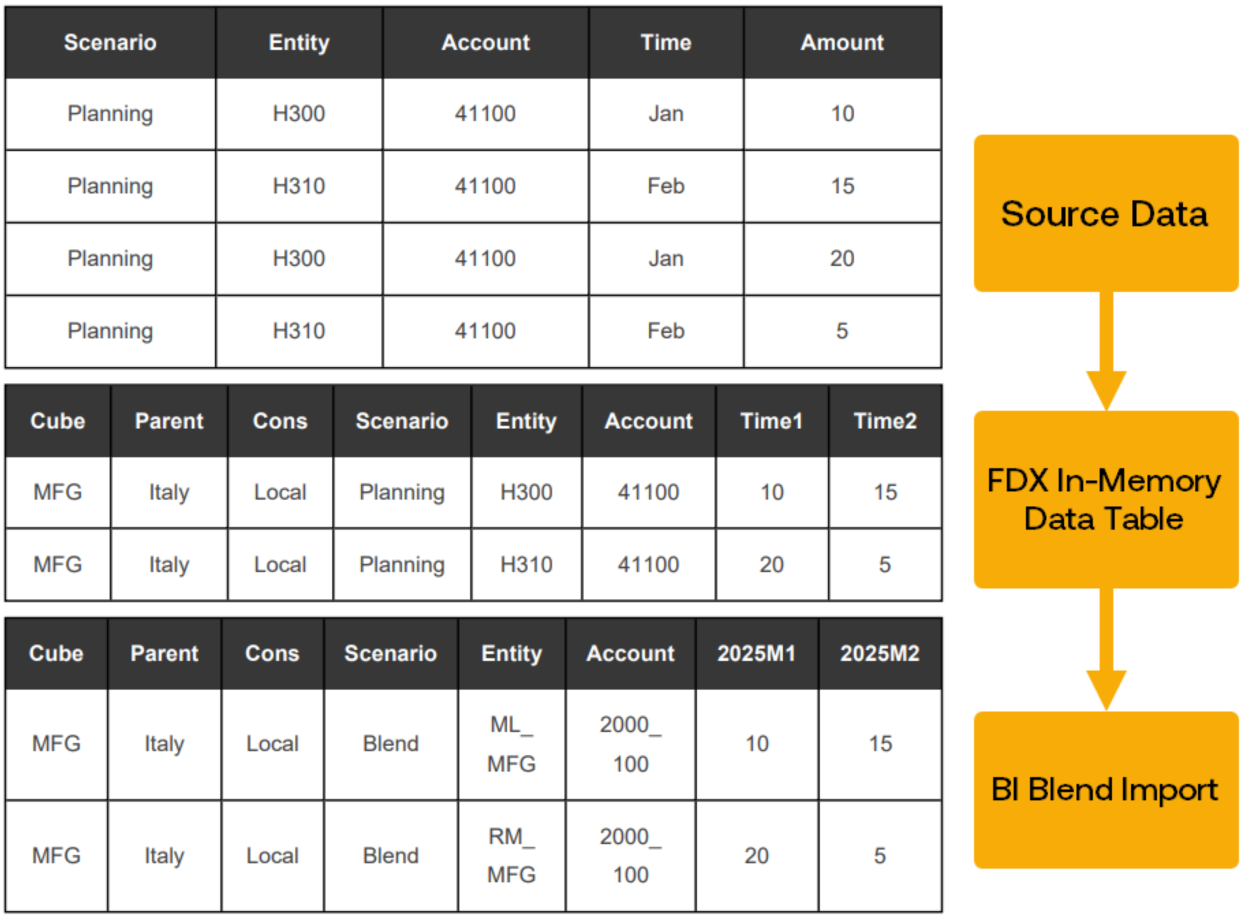
Entity	Scenario	Time	Account	Amount
E1	Actual	2025M2	Cash	100

Time as a Measure

Time is pivoted so that there are time columns for 2025M1 and 2025M2 to store the amounts.

Entity	Scenario	Account	2025M1	2025M2
E1	Actual	Account Receivable	150	250
E1	Actual	Gross Revenue	150	250
E1	Actual	Cash	50	100

The following image displays time pivot as part of the BI Blend import process. In this example, the warehouse FDX BRApi is used to extract data from the source.



Scenario	Entity	Account	Time	Amount
Planning	H300	41100	Jan	10
Planning	H310	41100	Feb	15
Planning	H300	41100	Jan	20
Planning	H310	41100	Feb	5

Cube	Parent	Cons	Scenario	Entity	Account	Time1	Time2
MFG	Italy	Local	Planning	H300	41100	10	15
MFG	Italy	Local	Planning	H310	41100	20	5

Cube	Parent	Cons	Scenario	Entity	Account	2025M1	2025M2
MFG	Italy	Local	Blend	ML_MFG	2000_100	10	15
MFG	Italy	Local	Blend	RM_MFG	2000_100	20	5

Source Data

In the source data table, Time is stored as a dimension, not as a measure. The time periods are in the Time column with the distinct time periods in rows. There is a separate Amount column to store the values.

FDX In-Memory Data Table

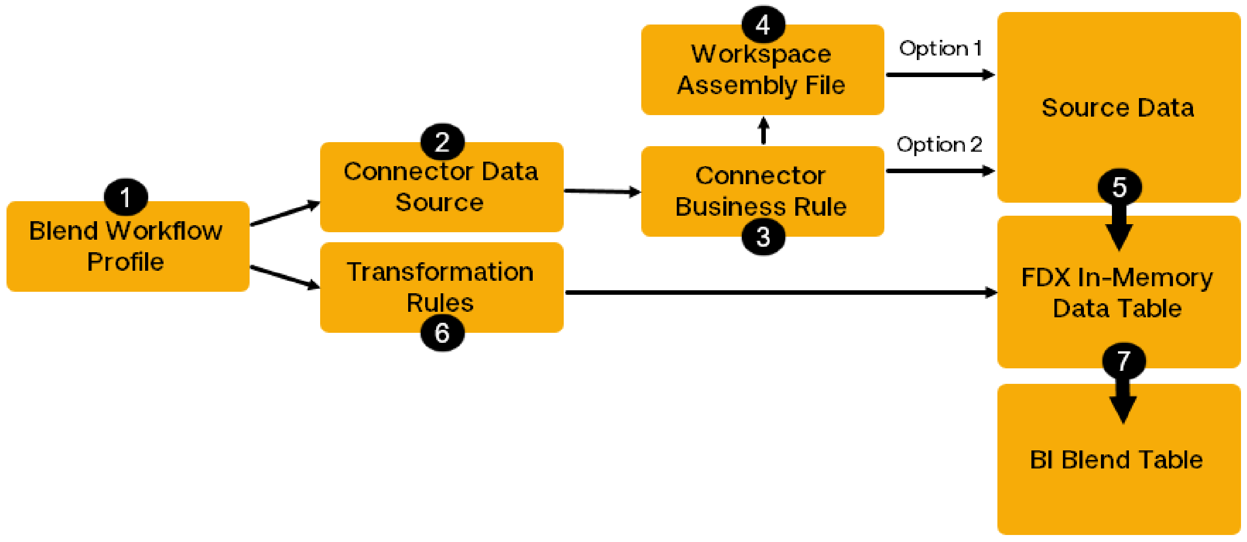
The FDX Warehouse BRApi extracts the data and stores it in an In-Memory data table. Time is pivoted so that Time is a measure. There are columns to store Jan and Feb amounts: Time1 and Time2. The column headers and members names are still the source member names. They have not been transformed yet.

BI Blend Table

The data is then imported into the BI Blend table where Time is also treated as a measure. The transformation rules are processed so column headers and member names show the target members and names. For example, Time1 is transformed to 2025M1, and Entity member H300 is transformed to ML_MFG. The data is stored against the Scenario dimension member that has been used to import the Blend data.

Set Up Fast Data Extract

The following diagram shows the import process and displays the artifacts that must be set up for FDX BRApis to extract data for import:



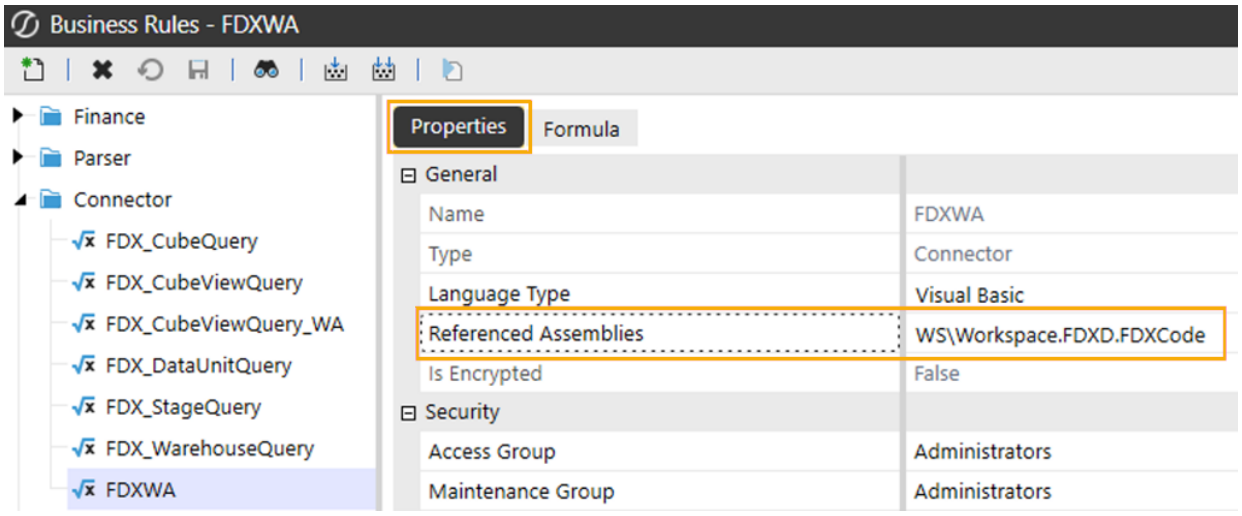
1. When a Load and Transform is run, the import calls the Blend Workflow Profile.
2. The Workflow Profile calls the connector data source.
3. The connector data source processes the connector business rule.

4. The connector business rule processes the Workspace Assembly file that contains the FDX BRApi, and data is extracted from the data source.
5. The extracted metadata and data are stored in the FDX In-Memory data table. If using a TimePivot API, time is pivoted.
6. Transformation rules are processed.
7. Data is imported into the BI Blend table using the BI Blend Settings defined in the Blend Workflow Profile.

Connector Business Rule

You can call the FDX BRApis directly in your connector business rule, or you can call them using a Workspace Assembly file. However, if the FDX BRApis are in a Workspace Assembly file, the data source still needs to reference a connector business rule.

To reference a Workspace Assembly file from a connector business rule, select the rule and navigate to the **Properties** tab. In the **Referenced Assemblies** field, enter the Workspace Assembly using this syntax: WS\Workspace.WorkspaceName.AssemblyName



General

Name	FDXWA
Type	Connector
Language Type	Visual Basic
Referenced Assemblies	WS\Workspace.FDXD.FDXCode
Is Encrypted	False

Security

Access Group	Administrators
Maintenance Group	Administrators

In the **Formula** tab, return the Workspace Assembly file itself using this syntax: WS\Workspace.WorkspaceName.AssemblyName.FileName

Considerations

Review these considerations when setting up the Blend import using FDX and a connector business rule:

- If there are zeroes in the Data Unit for some time periods, it is recommended that you turn zero suppression off for each time period in the data source.
- Enable all Attribute Value dimensions needed in the cube Integrations tab for the Scenario Type of your Blend scenario.
- If you are importing using the TimePivot BRApi, ensure that the Data Controls Measure Type setting is TimeWFViewAV. Ensure that your Data Controls content Type setting includes Attributes or Extended Attributes. See Data Controls Settings.
- If you do not see the correct connector fields in the data source, check the connector business rule or Workspace Assembly file. Make the correction, and refresh your application.
- Check that the transformation rules are mapping to the correct members, including the scenario members.
- If you are using Workflow Profile Substitution Text Settings variables, ensure they are correct, including the timeMemFilter variable. See Substitution Text Settings and Filtering Data
- If you are using the TimePivot BRApi, ensure the viewName variable value for the FdxExecuteDataUnitTimePivot function is set to YTD. If this is not set, you will receive this error: No valid DataKeys (Scenario / Time) found in data source.

- If you are using the TimePivot BRApi and have multiple time periods in the data source, map the Amount in the data source to one of the time periods. Otherwise, you will receive this error when you import the data: No valid DataKeys (Scenario / Time) found in data source.

Workspace Assembly File

If using a Workspace Assembly file, write it next. The rule needs to get the field lists as columns from the data source, then it must run the Fast Data Extract.

1. Select the FDX BRApi to return the field list. The field list column is used to populate the data source connector fields. The following FDX BRApis are available:
   - FdxGetCubeVieworDataUnitColumnList
   - FdxGetStageTargetColumnList
   - FdxGetWarehouseColumnList

   See Fast Data Extract BRApis.
2. After the FDX BRApi is selected, define the variable values.

   > **Example:** FdxGetCubeViewOrDataUnitColumnList(si, timeMemFilter, isTimePivot, useGenericTimeColNames)

   The following variables are used in this FDX BRApi example:
   - **timeMemFilter**: This determines the number of time periods that will be listed in the data source connector fields. You can return multiple time periods with a maximum of 36.
   - **isTimePivot**: If TimePivot FDX BRApis are being used to return the data in time columns so that time is treated as a measure, ensure that this is set to True.

- **useGenericTimeColNames**: If set to True, this will use GenericTimeColNames Time1-Time36 in the data source connector fields. If set to False, this will use the TimeColNames defined in the data source.

> **NOTE:** The FDX application BRApis that return field names to the data source from your application have the dimensions that need to be returned in the data source connector fields. Only the time periods must be defined when the field names are returned. The FDX warehouse BRApi that extracts data from a warehouse needs the column fields to return to the data source connector fields. These need to be defined in addition to the Time columns.

3. Select the relevant FDX BRApi to extract the data and prepare it for import using the BI Blend import. Define its variables. The following FDX BRApis are available:

   - FdxExecuteCubeView
   - FdxExecuteDataunit
   - FdxExecuteCubeViewTimePivot
   - FdxExecuteDataUnitTimePivot
   - FdxExecuteStageTargetTimePivot
   - FdxExecuteWarehouseTimePivot

   See Fast Data Extract BRApis

   > **TIP:** If you are extracting data for multiple periods, it is recommended that you use the TimePivot FDX APIs to accelerate the import. See Time Pivot.

Use Workflow Profile Substitution Text Settings in your rule to define variable values and make them unique to each Workflow Profile import. See Substitution Text Settings.

Data Source

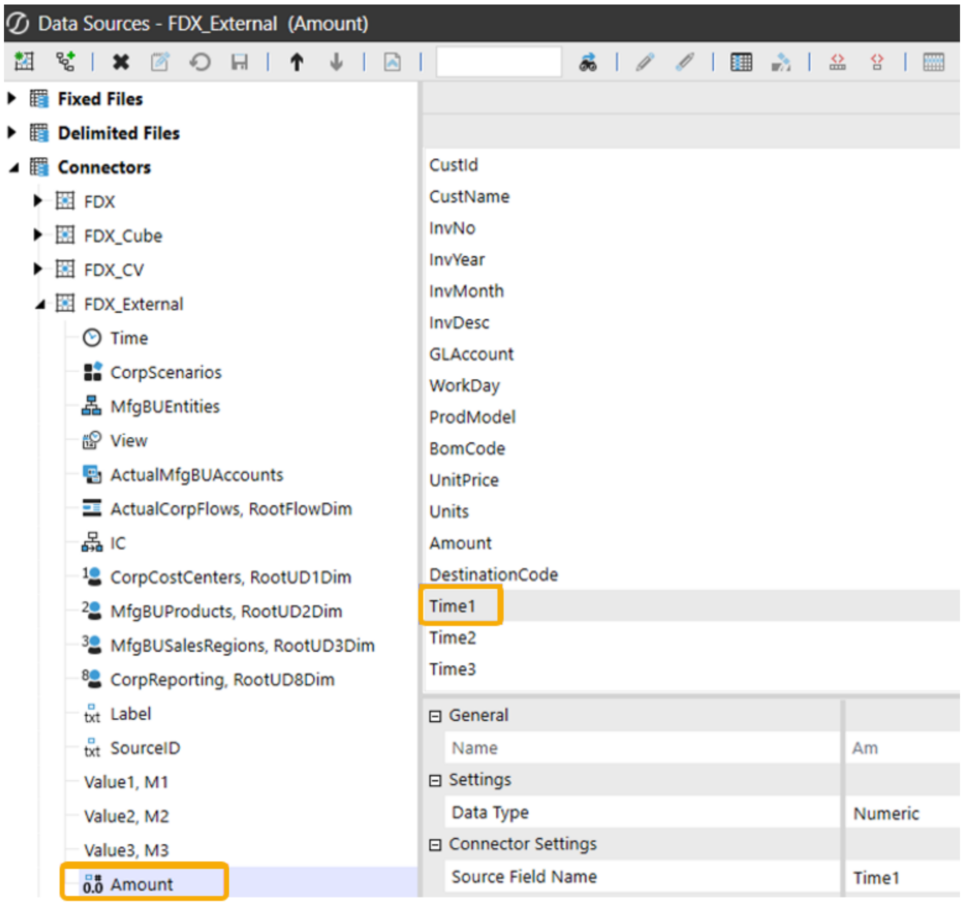
A data source must be set up to connect to the data table generated by the FDX extract.

> **NOTE:** If you are using the Time Pivot option, time is treated as a measure in your data table even if it is stored as a dimension in the source dataset. The time periods will be in columns across each record.

Complete the following steps to set up a data source:

1. Create a Connector Data Source and select your Connector Rule Name. See "Connector Data Source" in the *Design and Reference Guide*. The connector fields are returned by the FDXGet API used in your connector rule or your Workspace Assembly file.
2. Map the data source fields.
3. Assign the Time columns. If a Time Pivot API is being used, assign your time columns to your attribute value and extended attribute value dimensions if necessary.
4. If you are using a Time Pivot API, assign the Amount dimension to one of the Time columns. See Time Pivot.

   In this example, FdxExecuteWarehouseTimePivot is used to return data from an external warehouse data source to the FDX data table. Amount is mapped to the Time1 Source Field Name.


Data Sources - FDX_External (Amount)

- Fixed Files
- Delimited Files
- Connectors
  - FDX
  - FDX_Cube
  - FDX_CV
  - FDX_External
    - Time
    - CorpScenarios
    - MfgBUEntities
    - View
    - ActualMfgBUAccounts
    - ActualCorpFlows, RootFlowDim
    - IC
    - CorpCostCenters, RootUD1Dim
    - MfgBUProducts, RootUD2Dim
    - MfgBUSalesRegions, RootUD3Dim
    - CorpReporting, RootUD8Dim
    - Label
    - SourceID
    - Value1, M1
    - Value2, M2
    - Value3, M3
    - Amount

CustId
CustName
InvNo
InvYear
InvMonth
InvDesc
GLAccount
WorkDay
ProdModel
BomCode
UnitPrice
Units
Amount
DestinationCode
Time1
Time2
Time3

General	
Name	Am
Settings	
Data Type	Numeric
Connector Settings	
Source Field Name	Time1

Transformation Rules

Update the transformation rules as needed. See "Transformation Rules" in the *Design and Reference Guide*.

TIP: BI Blend only imports base members not parent members, so you may want to bypass these members.

Workflow Profile

Complete the following steps to set up a Blend Import Workflow Profile:

1. Assign the data source and transformation rules to your Blend Import Workflow Profile.
2. Define the BI Blend Settings. See Workflow Profile BI-Blend Settings

   NOTE: If Time Pivot is being used, select the relevant Content Type option to include the Attribute and Extended Attribute columns.

3. Define any variables that need to be defined in the relevant Substitution Text Settings field. See Substitution Text Settings.

Run the import and check the data in your BI Blend table to ensure you have the data needed.

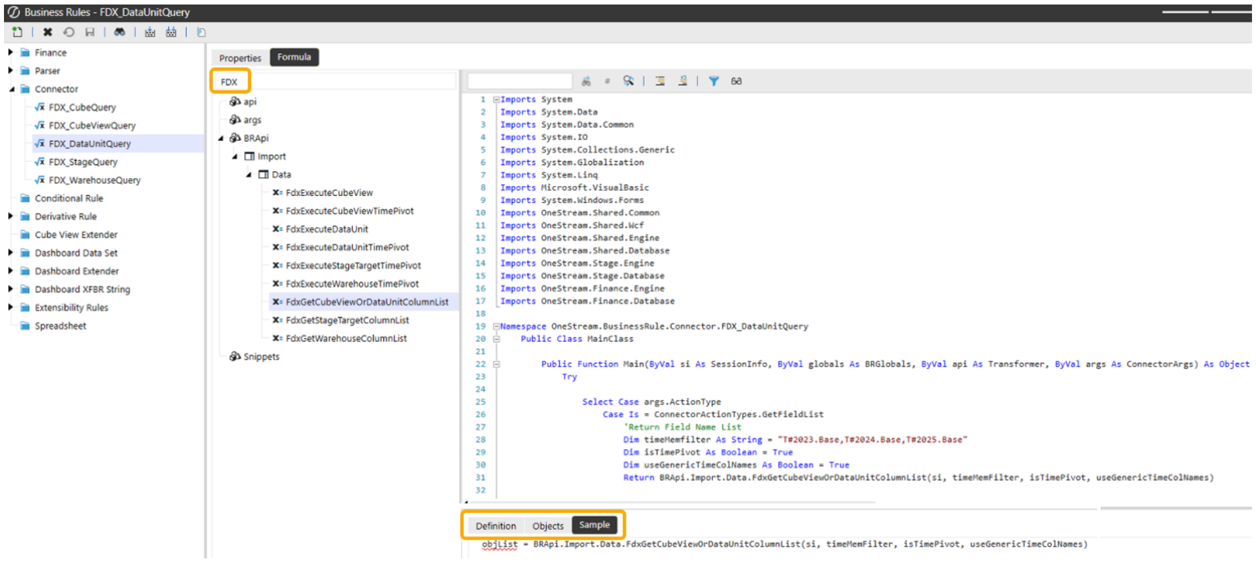
NOTE: BI Blend imports data for base members and aggregates the data. If you have parent member data in your Cube View or Data Unit, map those members to base members, or set an aggregation point so that BI Blend will aggregate the base data to parent members.

Fast Data Extract Sample Code

Each FDX BRApi function has its own variables that need values. Business rules and Workspace Assemblies provide function definitions and sample code to help users understand what variables each function needs and how they must be defined.

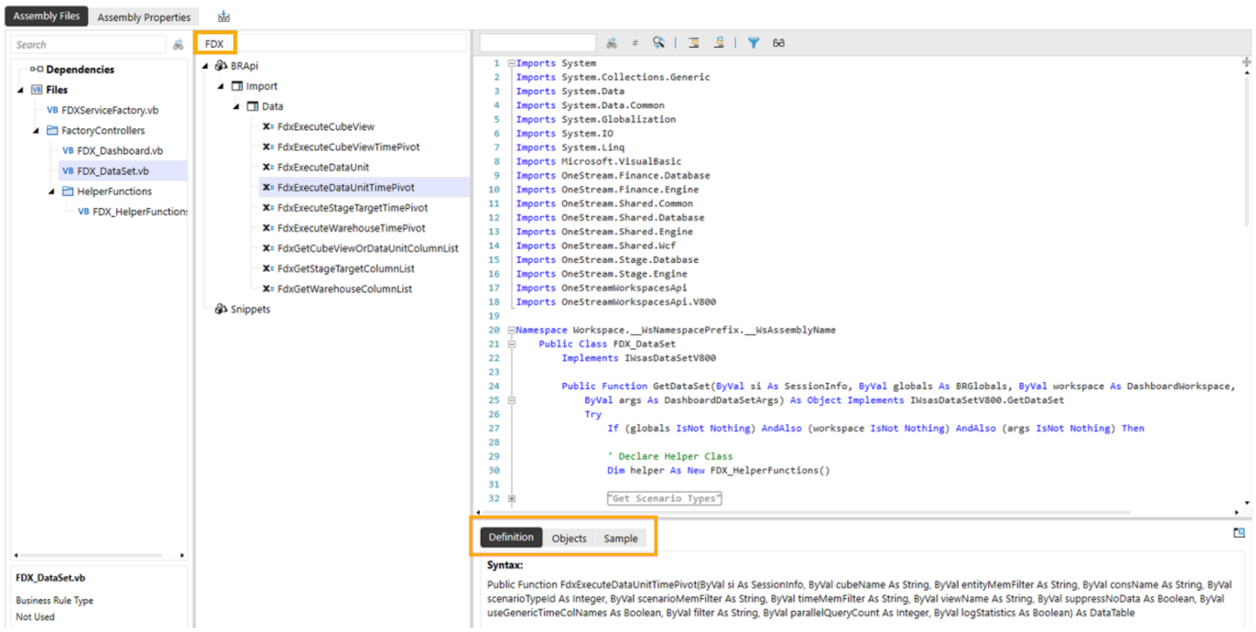
To view function definitions and sample code for business rules, select the business rule and navigate to the **Formula** tab. For Workspace Assemblies, select the assembly file, and navigate to the **Assembly Files** tab. Go to **BRApi** > **Import** > **Data** or enter **FDX** in the search bar. The Definition, Objects, and Sample tabs display:

- **Definition**: Displays how the variable must be defined, for example, As String or As Integer.
- **Objects**: Displays information about some of the objects themselves. You can select the object and view functions that can be used by that object.
- **Sample**: Provides sample syntax that can be copied and pasted.



```
Imports System
Imports System.Data
Imports System.Data.Common
Imports System.IO
Imports System.Collections.Generic
Imports System.Globalization
Imports System.Linq
Imports Microsoft.VisualBasic
Imports System.Windows.Forms
Imports OneStream.Shared.Common
Imports OneStream.Shared.Wcf
Imports OneStream.Shared.Engine
Imports OneStream.Shared.Database
Imports OneStream.Stage.Engine
Imports OneStream.Stage.Database
Imports OneStream.Finance.Engine
Imports OneStream.Finance.Database

Namespace OneStream.BusinessRule.Connector.FDX_DataUnitQuery
    Public Class MainClass

        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs) As Object
            Try

                Select Case args.ActionType
                    Case Is = ConnectorActionTypes.GetFieldList
                        'Return Field Name List
                        Dim timeMemfilter As String = "T#2023.Base,T#2024.Base,T#2025.Base"
                        Dim isTimePivot As Boolean = True
                        Dim useGenericTimeColNames As Boolean = True
                        Return BRApi.Import.Data.FdxGetCubeViewOrDataUnitColumnList(si, timeMemFilter, isTimePivot, useGenericTimeColNames)

```

```
objList = BRApi.Import.Data.FdxGetCubeViewOrDataUnitColumnList(si, timeMemFilter, isTimePivot, useGenericTimeColNames)
```



```
Imports System
Imports System.Collections.Generic
Imports System.Data
Imports System.Data.Common
Imports System.Globalization
Imports System.IO
Imports System.Linq
Imports Microsoft.VisualBasic
Imports OneStream.Finance.Database
Imports OneStream.Finance.Engine
Imports OneStream.Shared.Common
Imports OneStream.Shared.Database
Imports OneStream.Shared.Engine
Imports OneStream.Shared.Wcf
Imports OneStream.Stage.Database
Imports OneStream.Stage.Engine
Imports OneStreamWorkspacesApi
Imports OneStreamWorkspacesApi.V800

Namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
    Public Class FDX_DataSet
        Implements IWsasDataSetV800

        Public Function GetDataSet(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal workspace As DashboardWorkspace,
            ByVal args As DashboardDataSetArgs) As Object Implements IWsasDataSetV800.GetDataSet
            Try
                If (globals IsNot Nothing) AndAlso (workspace IsNot Nothing) AndAlso (args IsNot Nothing) Then

                ' Declare Helper Class
                Dim helper As New FDX_HelperFunctions()

                "Get Scenario Types"
```

Syntax:

Public Function FdxExecuteDataUnitTimePivot(ByVal si As SessionInfo, ByVal cubeName As String, ByVal entityMemFilter As String, ByVal consName As String, ByVal scenarioTypeId As Integer, ByVal scenarioMemFilter As String, ByVal timeMemFilter As String, ByVal viewName As String, ByVal suppressNoData As Boolean, ByVal useGenericTimeColNames As Boolean, ByVal filter As String, ByVal parallelQueryCount As Integer, ByVal logStatistics As Boolean) As DataTable

Substitution Text Settings

Each FDX function has its own set of variables and variable values that must be defined. These variable values can be hard coded in the rule, or the values can be provided in the Workflow Profile Substitution Text Settings.

You can do this by using the NameValueFormatBuilder. This function defines which of the Workflow Profile Substitution Text fields to use. Using the NameValueFormatBuilder function within your Workflow Profile Substitution Text Settings enables each workflow import's FDX API variable value to have a unique definition. After this function has defined which Text field to use, define which variable values will be referenced using the nvb.NameValuePairs.XFGetValue function.

To use the NameValueFormatBuilder, complete the following steps:

1. Declare the NameValueFormatBuilder and the Workflow Profile Substitution Text field being used.

   > **Example:** Dim nvb As New NameValueFormatBuilder (api.WorkflowProfile.GetAttributeValue (api.ScenarioTypeID, SharedConstants.WorkflowProfileAttributeIndexes.Text 4))

2. Declare the variable being defined and have it use the NameValueFormatBuilder variable. In the following example, the CubeViewName variable is being defined.

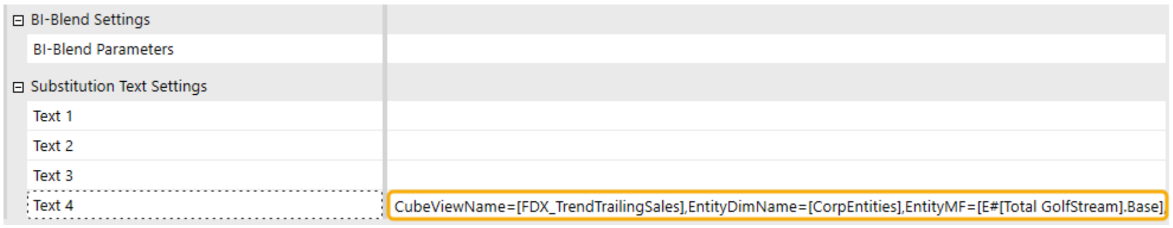
   > **Example:** Dim cubeViewName As String = nvb.NameValuePairs.XFGetValue ("CubeViewName", String.Empty)

```
Public Function Main (ByVal si As Session Info, ByVal globals As BRGlobals, ByVal api
As Transformer, ByVal args As ConnectorArgs) As Object
    Try
        Select case args.ActionType
            Case Is = Connector ActionTypes.GetFieldList
            'Return Field Name List
                Dim timeMemfilter As String = "T#2018.Base"
                Dim isTimePivot As Boolean = True
                Dim useGenericTimeColNames As Boolean = True
                Return BRApi.Import.Data.FdxGetCubeVieworDataUnitColumnList(si,
timeMemFilter, isTimePivot, useGenericTimeColNames)

        Case Is = ConnectorActionTypes.GetData
        'Parameter Format for WorkflowProfile Text4
        'EntityMF=[E#[Total GolfStream]. Base], ConsName=[Local], ScenarioTypeName=
[Actual], ScenarioMF=[S#Actual], TimeMF=[T#2017.Base], ViewName=[YTD], Suppress
NoData=[0], Filter=[Account Like '1** or Account Like '2** or Account Like '3** or
Account Like '4** or Acco

        'Get Parameters stored in Text4
        Dim nvb As New NameValueFormatBuilder (api.workflowProfile.GetAttributeValue
(api.ScenarioTypeID, SharedConstants.workflowProfileAttributeIndexes.Text4))
        Dim cubeName As String = api.CubeName
        Dim entityMemFilter As String = nvb.NameValuePairs.XFGetValue("EntityMF",
String.Empty)
        Dim parentName As String = nvb.NameValuePairs.XFGetValue("ParentName",
String.Empty)
        Dim consName As String = nvb.NameValuePairs.XFGetValue("ConsName",
String.Empty)
        Dim scenarioTypeName As String = nvb.NameValuePairs.XFGetValue
("ScenarioTypeName", ScenarioType.Actual.Name)
        Dim scenarioTypeId As Integer = ScenarioType.GetItem(scenarioTypeName).Id
        Dim scenarioMemFilter As String = nvb.NameValuePairs.XFGetValue("ScenarioMF",
String.Empty)
        Dim timeMemFilter As String = nvb.NameValuePairs.XFGetValue("TimeMF",
String.Empty)
        Dim viewName As String = nvb.NameValuePairs.XFGetValue("ViewName",
String.Empty)
        Dim suppressNoData As Boolean = ConvertHelper.ToBoolean
(nvb.NameValuePairs.XFGetValue("SuppressNoData", "1"))
        Dim useGenericTimeColNames As Boolean = True
        Dim filter As String = nvb.NameValuePairs.XFGetValue("Filter", String.Empty)
        Dim parallelQueryCount As Integer = 8
        Dim logstatistics As Boolean = False

        'Process Data
        Dim dt As DataTable = BRApi.Import.Data.FdxExecuteDataUnitTimePivot (si,
cubeName, entityMemFilter, consName, scenarioTypeId, scenarioMemFilter, timeMemFilter,
viewName, suppress NoData, useGenericTimeColNames, filter, parallelQueryCount,
logstatistics)
        If Not dt Is Nothing Then
            api.Parser.ProcessDataTable(si, dt, True, api.ProcessInfo)
        Else
            BRApi.ErrorLog.LogMessage(si, "FDX DU Params", nvb.GetFormatString())
        End If

        Return Nothing
```

3. In the Workflow Profile Substitution Text Settings, define the variable in the **Text 4** field. Use the syntax Variable=Value in a comma-separated list.

   Example: CubeViewName=[NameofCubeView]

   Example: CubeViewName=[Example],EntityDimName=[MfgBUEntities],EntityMF=[[E#EUR_MFG].Base],ScenarioDimName=[CorpScenarios],ScenarioMF=[S#Actual],TimeMF=[T#2025].base,Filter=[Unknown]



⊟ BI-Blend Settings

BI-Blend Parameters	

⊟ Substitution Text Settings

Text 1	
Text 2	
Text 3	
Text 4	CubeViewName=[FDX_TrendTrailingSales],EntityDimName=[CorpEntities],EntityMF=[E#[Total GolfStream].Base]

Filtering Data

FDX BRApis provide a filter option that enables you to specify a subset of the source dataset to extract. You can filter using source dataset member names and values. In addition to the entityMemFilter, scenarioMemFilter, and timeMemFilter, all the GetData FDX BRApis have the As String filter variable. This filter variable returns specific datasets.

The filter can be defined in the connector business rule or Workspace Assembly file itself. You can also use the NameValueFormatBuilder and define the filter in the Workflow Profile Substitution Text Settings fields. Where the filter is defined depends on whether you need a general filter used by different Workflow Profiles or if you need a filter specific to the Workflow Profile. For a filter specific to the Workflow Profile, use the Substitution Text Settings fields. See Substitution Text Settings.

IMPORTANT: The filter is searching for names and values in the source dataset before any transformation rules, including derivative rules, are applied.

TIP: When testing the filters, you can set your BI Blend Content Type to include source columns. This enables you to check the data being returned. See Data Controls Settings.

Filter Syntax

The syntax required is different depending on where the filter is defined. If the filter is defined in the rule itself, the string must be surrounded by quotation marks.

> **Example:** Dim filter As String = "Account Like '1*' or Account Like '2*'"

If the filter is defined in a Substitution Text Setting field, the string needs to be surrounded by square brackets.

> **Example:** Filter=[Account Like '1*' or Account Like '2*']

Follow these steps to build the filter syntax:

1. **Connector Field Column Name**: Define the Connector Field column name. The filter variable needs this name defined to determine which column to search on. Review the Connector Field column names returned in the data source to determine which column names to use. If you are using the TimePivot BRApis and you want to search by using values, reference the Time column name rather than using the Amount column name.

   > **Example:** Filter=[Account Like '*400' and Time1 > 50000]

2. **Like**: Specify the name of the member.
3. **And/Or**: You can use these to search for more than one connector field name and search criteria.

   > **Example:** Filter=[Account Like '*400' and Time1 > 5000 or Account Like '*300' and Time1 > 2000]

4. **Single Quotes**: If the search criteria is text, it must be surrounded by single quotes. If the search criteria is a value, it does not need to be surrounded by single quotes.

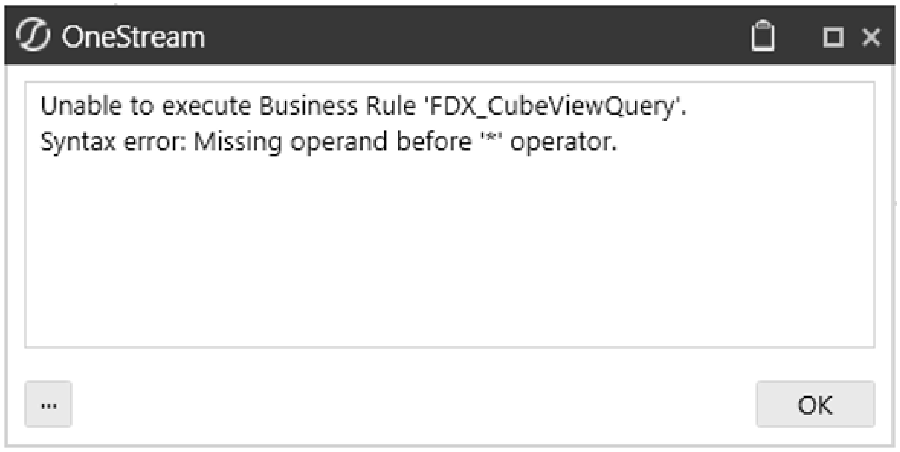
5. **Search Criteria**

   - **Names**: You can specify an exact name or use wildcard characters to search for multiple names, such as * or %.

   - **Values**: You can search for values that meet a specific criteria using standard visual basic operators, such as >< = .

> **NOTE:** If you do not want to use any search criteria in the filter, set it to String.Empty in the business rule. For example, Dim filter As String = String.Empty. If using the NameValueFormatBuilder to reference a filter in the Workflow Profile Substitution Text Settings, use either filter=[] or filter=[Unknown].

Syntax errors are returned in an error message as the data is imported. For example, this error message was returned during the Blend import because there were no single quotes around the search name criteria: Unable to execute Business Rule 'FDX_CubeViewQuery'. Syntax error: Missing operand before '*' operator.



```
Unable to execute Business Rule 'FDX_CubeViewQuery'.
Syntax error: Missing operand before '*' operator.
```

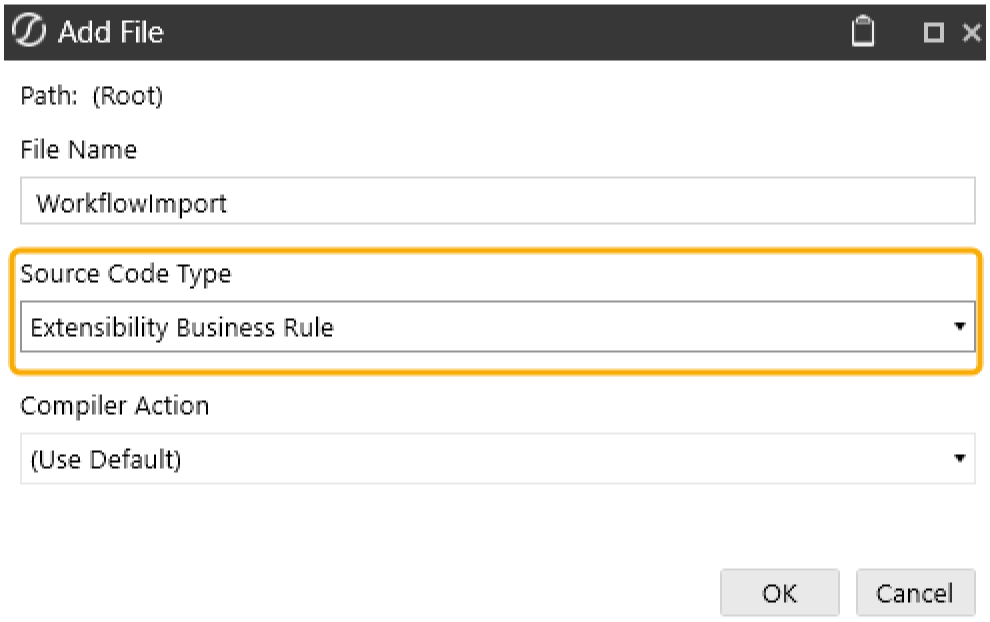
Import Automation

When a workflow import is scheduled, these steps are completed:

1. A Task Scheduler task calls a data management sequence.
2. The data management sequence calls a data management step.
3. The data management step calls a Workspace Assembly extender business rule file.
4. The Workspace Assembly file processes the import.

Complete the following steps to automate your import:

1. Create a Workspace Assembly extender business rule file. From the **Source Code Type** drop-down menu, select **Extensibility Business Rule**. See "Create an Assembly" in the *Design and Reference Guide*.



Path: (Root)

File Name: WorkflowImport

Source Code Type: Extensibility Business Rule

Compiler Action: (Use Default)

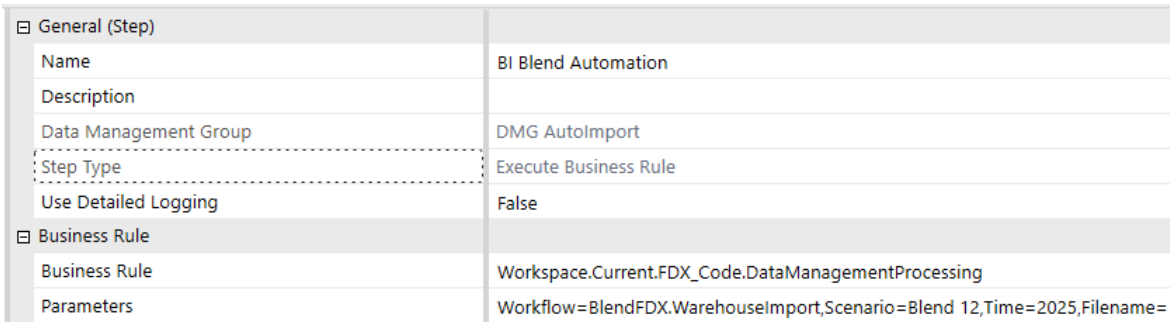
You can automate a BI Blend import using different data sources: Connector data source (Direct Load) and a delimited file data source.

> **NOTE:** For a delimited file data source, this BRApi references the workflow import Incoming folder, which is used to store the files that will be imported. To upload a file to the workflow import Incoming folder, navigate to **OneStream File Explorer** > **File Share** > **Incoming**, then expand the top-level workflow cube folder. Upload and import files to the relevant Workflow Profile folder.

Because the BI Blend import does not load data to a cube, you can use BRApi.Import.Process.ExecuteParseAndTransform to automate your import, rather than BRApi.Utilities.ExecuteFileHarvestBatchParallel. The variables and objects this BRApi expects can be referenced in the Definition, Objects, and Sample tabs. See Fast Data Extract Sample Code.

2. After the Assembly File is created, create a data management step. In the **Create Step** window, select **Execute Business Rule**. See "Build Data Management in Workspaces" in the *Design and Reference Guide*.

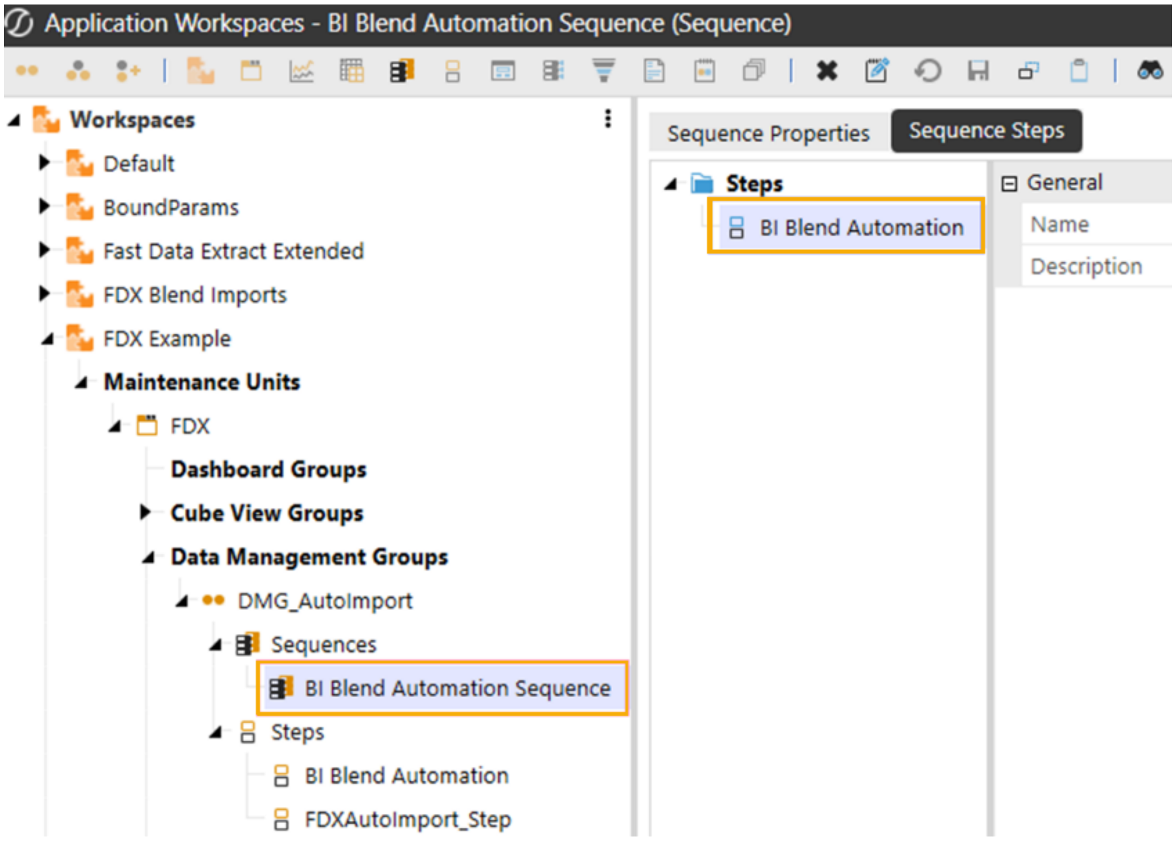
   Reference the business rule using this naming convention: Workspace.Current.MyWsAssemblyName.MyShortBRName



⊟ General (Step)	
Name	BI Blend Automation
Description	
Data Management Group	DMG AutoImport
Step Type	Execute Business Rule
Use Detailed Logging	False
⊟ Business Rule	
Business Rule	Workspace.Current.FDX_Code.DataManagementProcessing
Parameters	Workflow=BlendFDX.WarehouseImport,Scenario=Blend 12,Time=2025,Filename=

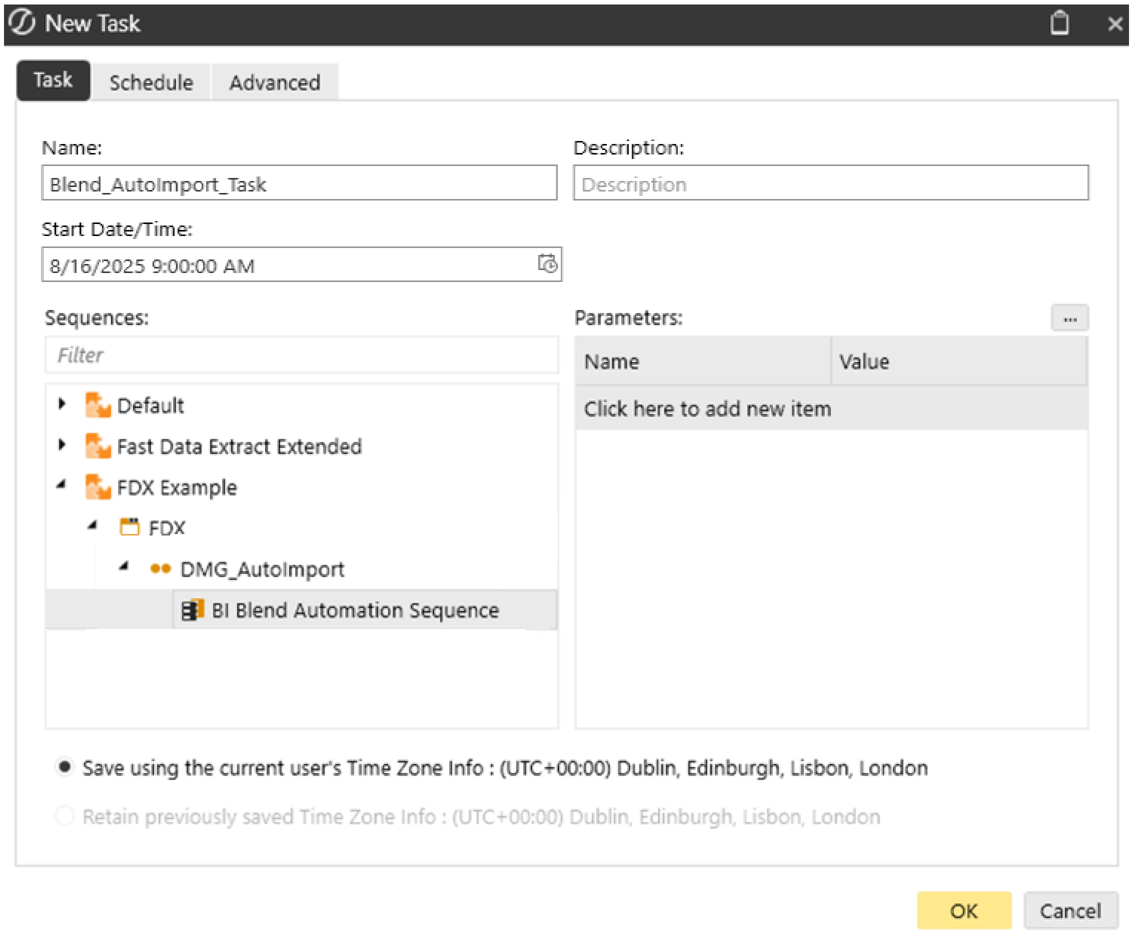
3. Create a data management sequence. A Task Scheduler task can only run data management sequences, so the step must be added to the sequence. See "Build Data Management in Workspaces" in the *Design and Reference Guide*.

   Test the data management sequence to ensure it runs successfully. You can also test each data management step separately.



TIP: You can set up email notifications and define who is notified for each event type. In the data management sequence you created, go to **Sequence Properties** > **Notifications**.

4. Create a Task Scheduler task to run the data management sequence at a specific date and time. In the **Task** tab, give the task a starting date and time, and select the data management sequence from the navigation tree. In the **Schedule** tab, set how often the task will be run. See "Task Scheduler" in the *Design and Reference Guide*.



Name: Blend_AutoImport_Task

Description: Description

Start Date/Time: 8/16/2025 9:00:00 AM

Sequences:

Filter

- Default
- Fast Data Extract Extended
- FDX Example
  - FDX
    - DMG_AutoImport
      - BI Blend Automation Sequence

Parameters:

Name	Value
Click here to add new item	

● Save using the current user's Time Zone Info : (UTC+00:00) Dublin, Edinburgh, Lisbon, London

○ Retain previously saved Time Zone Info : (UTC+00:00) Dublin, Edinburgh, Lisbon, London

Batch Harvest API

Instead of using the ExecuteParseAndTransform API, you can use this Batch Harvest API to automate your Blend workflow import: BRAPi.Utilities.ExecuteFileHarvestBatchParallel.

This references the Batch Harvest folder, which is used to store the files that are going to be imported. This API can be used with a Direct Load or File Import.

If you use the Batch Harvest API for a direct load, a trigger file must be created and saved to the Batch Harvest folder. The trigger file name tells OneStream which Workflow Profile to import data to. The trigger file can be created manually, or it can be generated by the extensible business rule. See "Batch File Name Format Specification" in the *Design and Reference Guide*.

Fast Data Extract Considerations

Review these considerations when using different FDX BRApis.

General Considerations

The BRApis used for extracting data from the data sources have these common variables:

- **useGenericTimeColNames (As Boolean)**: If you are pivoting Time and will be using Attribute Value dimensions, set your BI Blend MeasureType to TimeWFViewAV. In this case, set useGenericTimeColNames to True. If you want to determine Time using your source dataset, set useGenericTimeColNames to False.
- **filter (As String)**: You can filter the dataset to determine exactly which data from your source dataset to import. See Filtering Data .
- **parallelQueryCount (As Integer)**: It is best practice to set this to eight.
- **logStatistics (As Boolean)**: To log the statistics of your import to the error log, set this to True. If set to True, this generates a row in the error log called FDX Warehouse Query Statistics. These statistics include information about memory usage and the time it took to load each Blend Unit member in milliseconds.

Data Unit Considerations

The Data Unit FdxExecuteDataunitTimePivot BRApi enables you to return Data Unit data. You cannot extract dynamically calculated data with this option.

The variables available provide several options that enable you to define your Data Unit dimensions: entityMemFilter, scenarioMemFilter, and timeMemFilter.

> **NOTE:** You can extract parent entity data with this variable but, because BI Blend imports base member data only, you must map these to a base entity in your Transformation Profile.

Values for these variables must be declared: parentName, consName, scenarioTypeId, and viewName.

The consName variable enables you to extract data at the Local or Translated level only. You cannot extract Share or Elimination data.

The scenarioTypeId variable requires an ID returned from the name. To do this, return the name of the scenario and then return the ID of the name.

Cube View Considerations

Consider the following items when using the FDX BRApis to return Cube View data.

workspaceID

The variables for the FdxExecuteCubeViewTimePivot BRApi include a workspaceID. If you have Cube Views with the same name in different Workspaces, you must indicate which Workspace to use to find your Cube View.

TIP: The workspaceID can be returned using this function: Dim gValue As Guid = BRApi.Dashboards.Workspaces.GetWorkspaceIDFromName(si, isSystemLevel, workspaceName)

Cube View Point of View Row Headers

Consider how the FDX BRApi determines the Cube View members to understand the data that is returned. The FDX BRApi uses the rows in the Cube View to determine which members to return. Column dimension members are not used because there may be multiple members from the same dimension in the columns. The API will not be able to determine which of these column dimension members to use for the Cube View row.

The Cube View FDX BRApi will check the row headers of the Cube View to see which members to return. If dimension members are not defined in the rows, then the Cube View POV will be used to determine the dimension member. If this is not defined in the Cube View POV, then the Cube POV is used.

To check which members are returned, turn off the default row headers and decide which dimensions you want to be displayed in the Cube View Row header. Navigate to your Cube View and go to **Designer** > **General Settings** > **Header Overrides** > **Row Headers**.

NOTE: Because the Cube View FDX BRApi uses row headers to determine members, this may impact Cube View design. To extract data from existing Cube Views, you may need to pivot your dimensions from columns to rows to extract the correct members.

Dynamic Calculations

If there are dynamic calculations in the Cube View, the data returned as a result of the dynamic calculations in the Cube View can then be imported into the BI Blend table using the Cube View BRApi.

Stage Considerations

FdxGetStageTargetColumnList
The Time members are defined using startTimeName and endTimeName rather than using timeMemFilter, which is used by the Cube View and Data Unit FDX BRApis. Ensure that startTimeName and endTimeName include all the time periods needed to return the data source connector fields. Values must be returned for these variables: parentWFProfileName and scenarioName. These variables define the Workflow POV and are used to extract stage target data.

Warehouse Considerations

NOTE: If you are importing data from an external data source that is in a virtual private network (VPN), then Smart Integration Connector (SIC) is the recommended method for managing your external data source connections.

FDX APIs can extract data from internal or external warehouse data. You must specify the following information for these FDX APIs:

- The column names that must be returned
- The column and members to use as the Time dimension, so that the members can be used to pivot Time
- The column to use as the Amount or Measure column

FdxGetWarehouseColumnList
This FDX API is used to return the column names to the Data Source Connector fields. These are different from the application APIs because you need to specify the column names you want to return. To do this, you need the following variables:

- **dimColNameList As List(Of String)**: This variable requires a list of the fields you want to return to your Data Source Connector Fields, which can then be mapped to your application dimensions.
- **timePivotColNameList As List(Of String)**: This warehouse FDX API can only be used with the Time Pivot option. This variable requires a list of Time members that are in your warehouse. These members are returned as the Time columns in your Data Source Connector Fields.

Attributes

You can also map columns in your warehouse data source to Attribute dimensions. To do this, set your **includeAttributes** variable to **True**.

FdxExecuteWarehouseTimePivot

This is used to extract the data. You must complete these tasks to use this FDX API:

- Select the data using Select statements.
- Declare which column to use as the Time dimension column.
- Declare which column to use as the Amount or Measure column.

Select Data

Use these variables to build your Select statement:

- pageSelect As String
- pageFrom As String
- pageCriteria As String
- pageGroupBy As String
- pageOrderBy As String

Time Dimension Column

The timeDimColName variable is used to declare which column stores the time period members.

Measure (Amount) Column

The measureColName variable is used to declare which column stores the amounts.

ParrallelItemColName

The parallelItemColName variable is used to declare which column to use to select distinct values. These values are then run in parallel when the FDX API is processed, up to a maximum of eight values at a time. The number is determined by the parallelQueryCount variable. Use the same column in your Select Distinct statement that you use as the parallelItemColName value.

> **TIP:** It is recommended that you select the column with the fewest distinct values for better performance.

Reporting Solutions

You can report on Blend data directly using OneStream SQL-based reporting options. Alternatively, you can report on Blend data using cube-based options by layering Dynamic Cube Services on top of the Blend model to remodel data as a dynamic cube.

Decide on a reporting solution by considering how much control you want over the filtering and display of data. For example, if you want the ability to create custom calculations on fields, you could use a Pivot Grid or Large Data Pivot Grid depending on data volume. Consider what needs to be reported on, including whether to include additional records generated by Derivative Rules and aggregation during a Blend import.

Component	End-user Control in a Dashboard	Data Visualization
Data Returned by: Component		
BI Viewer	Enables filtering	BI Viewer components. See the *BI Viewer Guide.*
Pivot Grid	Enables control of display	Grid only. See the *Pivot Grids Guide.*
Data Returned by: Data Adapter		

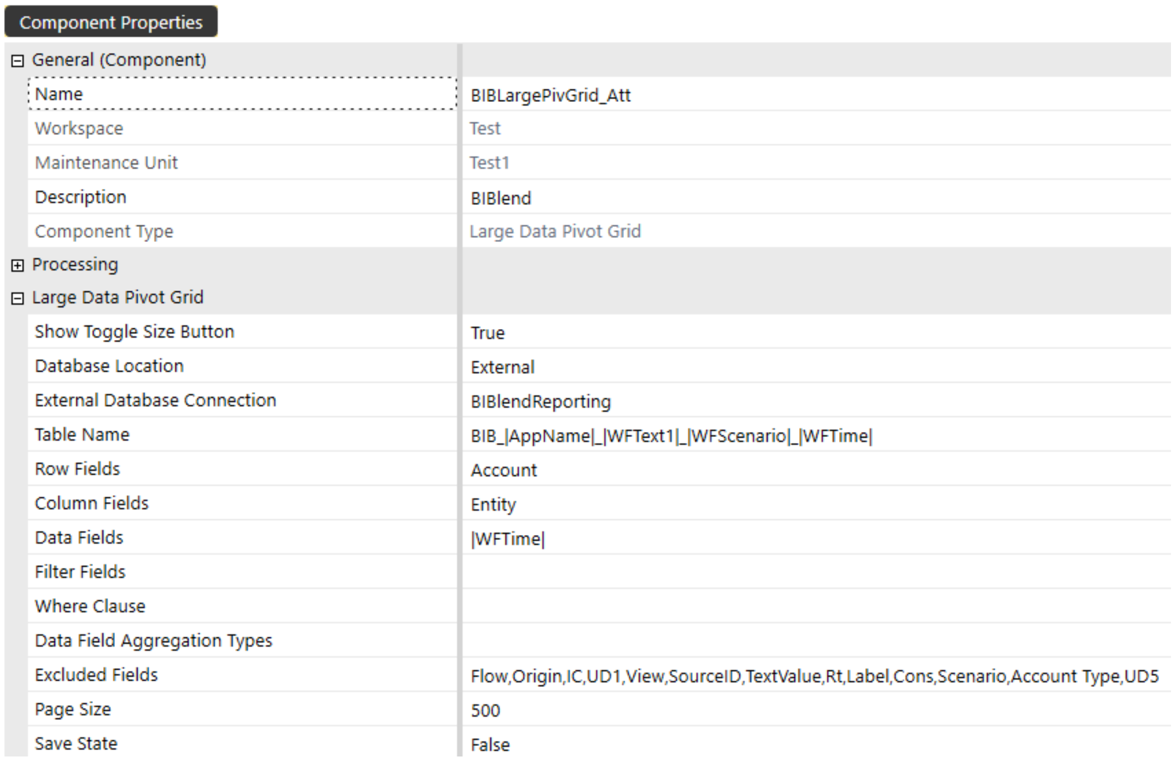
Component	End-user Control in a Dashboard	Data Visualization
Large Data Pivot Grid	Enables control of display, filtering, and calculation creation	Grid only. See "Large Data Pivot Grid" in the *Design and Reference Guide*.
SQL Table Editor	Enables filtering and editing of records	Grid only. See "SQL Table Editor" in the *Design and Reference Guide*.
Data Returned by: Dynamic Cube Services		
Cube View	Enables filtering and editing of records	Grid, Reports, Excel Add-In, and Dashboards (Cube View or Cube View MD Data Adapters). See "Dynamic Cube Services" in the *Design and Reference Guide*.
Excel Add-In	Enables control of display and calculations, calculation creation, filtering, and editing of records	Quick View, XFGetCells, Cube View, and Excel charting options. See "Dynamic Cube Services" in the *Design and Reference Guide*.

Large Data Pivot Grid

The Large Data Pivot Grid is a dashboard component designed to enable pivot style dashboard reporting and analysis on external database tables. See "Large Data Pivot Grid" in the *Design and Reference Guide*.

These are the key features of this component:

- Server based processing. This is a key solution for accessing data in large tables. Pivoting requests are performed on the server, returning only the requested slice of data.
- Supports paging to manage large datasets.
- Measures support only a single aggregation type (Sum, Min, Max).



⊟ General (Component)	
Name	BIBLargePivGrid_Att
Workspace	Test
Maintenance Unit	Test1
Description	BIBlend
Component Type	Large Data Pivot Grid
⊞ Processing	
⊟ Large Data Pivot Grid	
Show Toggle Size Button	True
Database Location	External
External Database Connection	BIBlendReporting
Table Name	BIB_|AppName|_|WFText1|_|WFScenario|_|WFTime|
Row Fields	Account
Column Fields	Entity
Data Fields	|WFTime|
Filter Fields	
Where Clause	
Data Field Aggregation Types	
Excluded Fields	Flow,Origin,IC,UD1,View,SourceID,TextValue,Rt,Label,Cons,Scenario,Account Type,UD5
Page Size	500
Save State	False

Data Adapters

Data Adapters with the following Command Types can be used to return data from Blend sources: SQL, BI-Blend, and Method.

> **NOTE:** The Command Types Cube View and Cube View MD only work with data modeled as a cube. These options are used if blend data is first pulled into a dynamic cube using Dynamic Cube Services.

See "Data Adapters" in the *Design and Reference Guide*.

SQL Data Adapter

This Data Adapter can be used to query any SQL database. The returned records do not need to be unique.

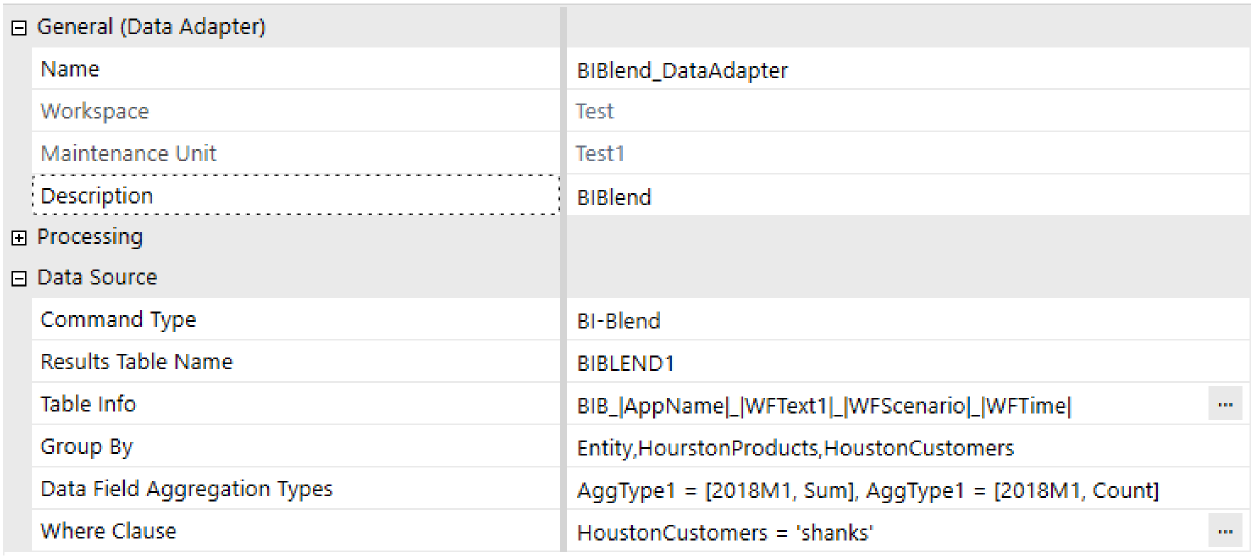
To display data from the StageBiBlendInformation table, your Database Location selection must be Application. To display data from your Blend tables and views, your Database Location selection must be External.

BI Blend Data Adapter

This Data Adapter is used to simplify writing queries to the BI Blend tables by eliminating the need for SQL scripting. This Data Adapter includes grouping, aggregation, and filtering fields. These fields are used to generate the SQL query. The BI Blend Data Adapter can only return unique records to the data table.

As an adapter, it cannot support the complete contents of very large BI Blend tables. The BI Blend Data Adapter should contain where clauses to slice the results into reporting slices. The BI Blend Data Adapter does not support paging to manage large volumes of records.

See "Data Adapters" in the *Design and Reference Guide*.



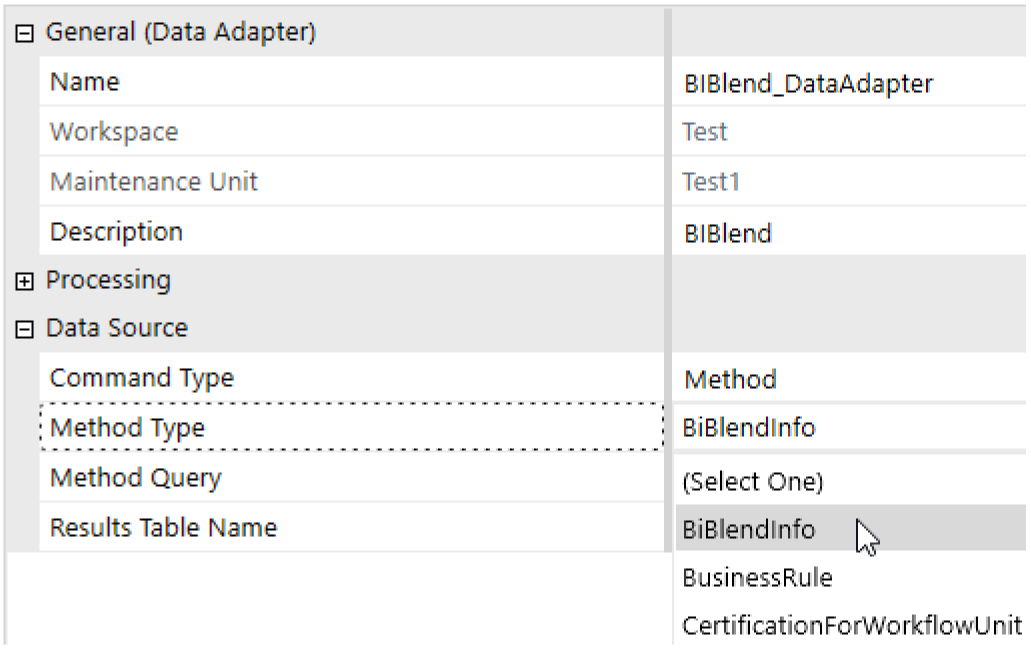
⊟ General (Data Adapter)	
Name	BIBlend_DataAdapter
Workspace	Test
Maintenance Unit	Test1
Description	BIBlend
⊞ Processing	
⊟ Data Source	
Command Type	BI-Blend
Results Table Name	BIBLEND1
Table Info	BIB_|AppName|_|WFText1|_|WFScenario|_|WFTime|
Group By	Entity,HourstonProducts,HoustonCustomers
Data Field Aggregation Types	AggType1 = [2018M1, Sum], AggType1 = [2018M1, Count]
Where Clause	HoustonCustomers = 'shanks'

Method Data Adapter

Method Data Adapters use predefined queries to return data. The Method Type options relevant to blend are BIBlendInfo and BusinessRule.

- **BiBlendInfo**: This method query only returns data from the application table StageBiBlendInformation. The query takes the following form: {myWorkflowProfileName}{ScenarioName}{WorkflowTimePeriod}{<Where Clause>}. The first three parameters define the BIB-Table name and are used to return imports that have been run for that table from the StageBiBlendInformation table. The last parameter defines the <Where Clause> and should take the form of Name = Value.

  NOTE: This method query returns information about a specific blend import. To return information on multiple blend imports, use a SQL Data Adapter.



⊟ General (Data Adapter)	
Name	BIBlend_DataAdapter
Workspace	Test
Maintenance Unit	Test1
Description	BIBlend
⊞ Processing	
⊟ Data Source	
Command Type	Method
Method Type	BiBlendInfo
Method Query	(Select One)
Results Table Name	BiBlendInfo
	BusinessRule
	CertificationForWorkflowUnit

- **BusinessRule**: This method query enables you to return data from your Blend tables and views by configuring a predefined query that triggers logic stored in a Dashboard Data Set Business Rule. This option enables you to enhance the data returned, for example, by pivoting fields or performing dynamic calculations. This logic should be stored in either a Dashboard Data Set Business Rule or a Data Set Service Workspace Assembly File.

Dynamic Cube Services

Dynamic Cube Services enables you to better integrate your stored and Blend data for cube-based reporting options. It can also be used to give dimensionality to transient data fields by implementing the Dynamic Dimension Service. In addition, if modeled as a dynamic cube, you can perform translations and consolidations on your Blend data. See "Dynamic Cube Services" in the *Design and Reference Guide*.

Review the following considerations:

- Extended blend attributes are not currently supported by Dynamic Cube Services.
- Dynamic Cube Services understands data as YTD. If your data is periodic, you may need to generate YTD data before using Dynamic Cube Services. Use a Blend Derivative Rule and TimeHelper function to produce records with YTD values.
- Blend tables can be large and the datasets sparse, so when layering Dynamic Cube Services on top of blend, target only the required records.
- Dynamic Cube Services does not support xBlend dimensions, such as xBlendDatetime.

Common Error Messaging

If a matching base member cannot be found in the data record and no data intersections are found, you may receive this error: BI-Blend base data does NOT match any Aggregation Base Members.

NOTE: A BI Blend data source cannot be used to import data to Stage because the extra BI Blend attributes and date dimensions do not work for Stage. The following error message is an example of a message you may receive: The given ColumnMapping does not match up with any column in the source or destination.

You will see the following error message when loading to a workflow if attribute value dimensions have been assigned a static value: No valid DataKeys (Scenario/Time) found in data source. Review the source data load processing log and check one-to-one transformation rules to ensure that you have created proper Scenario and Time dimension rules for this data source.

Error Messages and Suggested Solutions

Blend Import Error Message	Where	Solution
No valid DataKeys (Scenario/Time) found in data source.	Data load processing log	Review to find possible information on the source of the error.
	Data Source	Ensure the Amount field is mapped to a data column.
	Transformation Rules	Fix any transformation errors for all dimensions.
	Business Rules (FDX-based imports)	Ensure the viewName variable is set to YTD.
	Workflow Profile > BI Blend Settings	Ensure the appropriate Measure Type is selected. For example, TimeWFViewAV when using Attribute Values to import multiple period data.
Conversion from string “” to type ‘Date’ is not valid	Data Source	Ensure the format of the Blend datetime fields are set to MM/dd/yyyy hh:mm:ss.

Blend Import Error Message	Where	Solution
A column named ‘columnname’ already belongs to this DataTable	Cube > Cube Integrations	Check attribute aliases and column aliases to ensure attribute dimensions have unique names.
	Workflow Profile > Bi Blend Settings	
Cannot find column [columnname]	Workflow Profile > Bi Blend Settings	Check the Content Type setting to ensure that the column has not been excluded.
Rule [buffername] contains an invalid complex expression [details]	Derivative Rules	Use the details provided in the error message to help troubleshoot.
SQL Syntax Errors	Error Log	Write your SQL statement to the error log, review it and, if needed, test it using a SQL Data Adapter to see what syntax errors are returned.

Blend Import Error Message	Where	Solution
BI-Blend base data does NOT match any Aggregation Base Members	Workflow Profile > BI Blend Settings	Check whether aggregation has been set on a base member that doesn’t appear in the data records being imported.
	Transformation Rules	Check the base member is being transformed correctly.
Unable to execute Business Rule [Rule name]. Syntax Error: Missing operand before the ‘*’ operator	Business Rule	Check the FDX Filter syntax.
	Workflow profile > Text fields	Check that text fields are surrounded by single quotes.
The given ColumnMapping does not match up with any column in the source or destination	Dimension Library > Scenarios >ScenarioType	Make sure your Blend Scenario Type is only used by Blend scenarios.

Batch Processing Support

Batch Processing is supported through automating the BI Blend processusing Data Management and Task Scheduler functionality.