

Understanding the Financial Model Guide

9.4.1 Release

Copyright © 2026 OneStream Software LLC. All rights reserved.

All trademarks, logos, and brand names used on this website are the property of their respective owners. This document and its contents are the exclusive property of OneStream Software LLC and are protected under international intellectual property laws. Any reproduction, modification, distribution or public display of this documentation, in whole or part, without written prior consent from OneStream Software LLC is strictly prohibited.

Table of Contents

About the Financial Model 1

Data Units 2

Level 1: Cube Data Unit 2

Level 2: Workflow Data Unit 3

Level 3: Workflow Channel Data Unit 4

Workflow Channel Unit Budget Example 5

Data Loading 6

Data Load Execution Steps (Clear and Replace) 6

Workflow Profile Data Loading Behaviors 8

Data Locking 12

Lock Types 12

Security 15

Extensible Dimensionality 16

Recommended Implementation of Extensibility 20

Cube Design Choices within an Application 26
Application Model Design Considerations 27
Standard Cube Design 28
Guidelines 28
Performance Profile 29
Reporting Guidelines 29
Standard Large Sparse Cube Design 29
Design Guidelines 29
Performance Profile 30
Reporting Guidelines 30
Hybrid Cube Design 30
Guidelines 31
Performance Profile 31
Reporting Guidelines 32
Relational Blending 33
Relational Blending Types 33

Model Blending 34
Use Case 1 35
Use Case 2 35
Model Blending Benefits 36
Relational Blending API 37
Cache Level Types 38
Relational Model Blending API Functions 39
GetStageBlendTextUsingCurrentPOV 39
Parameters 39
Parameters 43
Parameters 45
Parameters 47
Parameters 48
Parameters 50
Parameters 52

BI Blend 53

Key Elements of BI Blend 53

Dimensions Overview 55

Configurable Dimensions 57

Dimension Library Navigation 57

Members Tab 60

Member Properties and Relationship Properties Tabs 61

Dimension Properties Tab 66

Grid View Tab 68

Build a Dimension 71

Create a Member 71

Clone a Member 71

Change the Position of a Member 72

Expand All Descendants 72

Copy and Paste Relationships 73

Add Relationship for Selected Member 73

Remove Relationships 74
Delete a Member 74
Entity Dimension 79
Scenario Dimension 88
Account Dimension 104
Flow Dimension 117
User Defined Dimensions 1–8 124
Parent Dimension 134
Intercompany (IC) Dimension 136
Time Dimension 138
Consolidation Dimension 142
System-defined Dimensions 145
Origin Dimension 145
View Dimension 147
Entity Aggregation 150
How Entity Aggregation Works 150

Example of an Entity Aggregation 151
Aggregated Dimension 152
Example of Cube View Using C#Aggregated Dimension 153
Launching an Aggregation 155
Entity Aggregation Input 156
Enable Entity Aggregation Input 156
Working With Hybrid Scenarios 157
Data Bindings 158
Data Filters 159
Share Data From the Source Scenario 160
Copy Data From the Source Scenario 160
Time Profiles 162
Anatomy of a Time Profile 163
Time Profile Toolbar 163
Profile Tab 164
Time Periods Tab 168

Default Year 168
Half Years 169
Quarters 170
Months 170
Weeks 171
Cubes 173
Cubes 173
Structure of a Cube 174
Cube Properties 174
Cube Dimensions 189
Cube Reference 202
Data Access Tab 204
Integration 212
Foreign Exchange Rates 228
Grid Settings Intersection 228
FX Rate Type 228

Rows and Columns 229
Lock FX Rates 230
Non-administrators: 230
Security Roles that Manage the Locking Features 231
To access FX Rates Security Roles 231
System Changes: Administrator Access to Lock FX Rates 232
View Task Activity 232
Lock and Unlock Rates 233
Consolidation 237
Example of a Consolidation 239
Launching a Consolidation 240
Calculation Status State 1: Actual Scenario, Loading for December 241
Calculation Status State 2: Plan Scenario, Loading for 12 Months 242
Calculation Status 243

Currency Translation 244

How the Origin Dimension Works with the Consolidation Dimension 246

Imports and Forms 246

Adjustments 247

Eliminations 248

Intercompany Elimination 249

Direct and Indirect Eliminations 252

Custom Consolidation 253

Matrix Consolidation: Eliminating Beyond Legal Entity 254

Entity Dimension or User Defined Dimension 254

Out-of-the-box View of Eliminations 262

Matrix Consolidation - View of Eliminations 264

Setup 265

Business Logic 268

Org-by-period in the User Defined Dimension 273

Equity Pickup 274
Equity Pickup Example 275
Filter Runtime Data and Reduce Maintenance with Parameters 277
About Parameters 277
Parameter Types 279
Literal Value Parameters 280
Input Value Parameters 280
Delimited List Parameters 281
Bound List Parameters 283
Member List Parameters 283
Member Dialog Parameters 284
Benefits 284
Ways to Use Parameters 285
Cube Views 286
Data Adapters and Dashboard Components 286
Extensible Documents 287

Books 287

Requirements 288

Definition and Assignment 288

Syntax 288

Best Practices 289

Design 289

Naming 291

About Parameter Definitions 291

Property Types 291

Create a Parameter 292

Specify Literal Value Settings 294

Specify Input Value Parameter Settings 295

Specify Delimited List Settings 295

Specify Bound List Settings 296

Specify Member List Settings 301

Specify Member Dialog Settings 302

Sample Delimited List Parameter in a Cube View 305

Manage Parameters 307

Assign a Parameter 307

Modify a Parameter 311

Copy a Parameter 311

Delete or Remove a Parameter 312

Rename a Parameter 313

Identify Parameter Assignments 314

Nest Parameters 314

About the Financial Model

The financial model includes extensible dimensionality, which allows you to extend the dimensions to suit your own financial purposes. Cubes and design model considerations, along with the use of an aggregate storage model called BI Blend, which supports large-volume reporting for data that is not appropriate to store in a traditional cube, allow you to further customize financial data modeling. Finally, hybrid scenarios help to improve data query performance when analyzing smaller sets of data that contain high volumes of account-level detail.

These tools, along with other data modeling techniques, allow you to build robust financial data models.

Data Units

A data unit is used to load, clear, calculate, store, and lock data in the multi-dimensional engine. With workflow channels, OneStream provides the following data units.

Level 1: Cube Data Unit

This is the largest unit of work and is commonly referred to as the entity, scenario and time data unit. This aligns with financial analytic system tasks to clear, load, calculate, and lock entity, scenario and time combinations.

Members of the cube data unit are:

- Cube
- Entity
- Parent
- Consolidation
- Scenario
- Time

Cube data unit analytic work items include (applicable to cube data processing):

- Clear data
- Load data
- Lock data

- Copy data
- Calculate
- Translate
- Consolidate

> **NOTE:** The Cube data unit is locked when an input parent workflow profile is locked or certified, provided an entity is assigned to this parent workflow step. See "Data Locking" in the *Design and Reference Guide* for more information.

Level 2: Workflow Data Unit

The workflow data unit builds on the cube data unit by including the account dimension. This decreases the unit of work by increasing the data unit's granularity. This indicates a workflow data unit is a sub-set of the cube data unit, allowing fine-grained control over analytic work items.

The workflow data unit is the default level used by the workflow engine to control, load, clear, and lock data during the processing of stage data. Workflow level data loads from the staging data mart to the cube, and is processed at a granular level that includes the account dimension by default.

For example, if two import workflow profiles are not siblings of the same input parent, but load to the same entity, scenario and time dimensions, the data loads and clears at the account level. However, if these two workflow profiles load the same accounts, the last workflow profile to load is used. If these workflow profiles load to different accounts, then data loads for both workflow profiles.

Members of the workflow data unit are:

- Cube
- Entity
- Parent
- Time
- Consolidation
- Scenario
- Account

Workflow data unit analytic work items are (applicable to stage data processing and cube data import):

- Clear data
- Load data
- Lock data

At the cube level, data is locked on the assigned entities by origin dimension and is dependent on the workflow step that is locked. This can be forms, import, or adjustment. See "Data Locking" in the *Design and Reference Guide* for more information.

Level 3: Workflow Channel Data Unit

The workflow channel data unit builds on the workflow data unit by including a single user-defined dimension. The user-defined dimension decreases the unit of work by increasing the data unit's granularity. This means a workflow channel data unit is a subset of the workflow data unit, allowing detailed control over analytic work items.

The user-defined dimension that extends the data unit is specified at the application level from the **Application Properties** screen. You can only use one user-defined dimension per application as a workflow channel, so carefully consider which user-defined dimension to select in relation to the application's account dimension.

For example, cost center and version user-defined dimensions are commonly used in a workflow channel data unit. These user-defined dimension are frequently included in a workflow channel data unit because they represent data slices that align with data collection and locking requirements of the budget and forecast business processes.

> **NOTE:** Accounts can also utilize workflow channels.

Workflow Channel Unit Budget Example

A typical cost center budget collection process has many users submitting data to a single cube data unit. In addition, all users are submitting data for different cost centers to the same workflow data unit (that is, the same account). This can inevitably cause data contention within the workflow data unit. Therefore, assigning the user-defined dimension containing cost centers as the user-defined dimension type for Workflow Channels makes each entity/cost center act as a granular and autonomous cell collection in a legal entity. This allows individual cost centers to load, clear, and lock data with no impact to other cost centers.

Workflow channel data unit members are:

- Cube
- Entity
- Parent
- Time
- Consolidation
- Scenario
- Account
- User Defined (x)

Workflow channel data unit analytic work items are (applicable to stage data processing and cube data import):

- Clear data
- Load data
- Lock data

At the cube level, data is locked on the assigned entities by workflow channel if assigned correctly. This applies to workflow channels used in the account or a user-defined dimension. See Data Locking.

Data Loading

This section describes how the workflow engine loads data for each relationship between the workflow and data units. The OneStreamworkflow engine controls data loading from the staging data mart to the analytic model. The workflow engine includes intelligence about what data to load and how to load for each workflow unit. The loaded analytic cube gets this from the binding relationship between its input parent workflow profiles and base entities.

In addition, the workflow engine uses the origin dimension's import member exclusively when loading data. This predefined relationship provides a built-in level of data protection between imported data, manual data entry, and journal adjustments. The workflow engine manages how data is placed into the origin dimension's import, forms, and adjInput members. The workflow engine also forces imported data to use the local member of the consolidation dimension.

The workflow engine always starts with a workflow data unit to control clearing, loading, and locking data for its entities. A workflow channel data unit is used if workflow channels are active in the workflow unit's analytic model relationship.

Data Load Execution Steps (Clear and Replace)

When a workflow unit's data load process executes, the engine does the following:

1. Checks workflow state
    - Implicitly locked (parent workflow certified)
    - Explicitly locked
2. Checks workflow profile data load switches
    - Can load unrelated entities (True / False)
    - Flow type no data zero view override (YTD / Periodic)
    - Balance type no data zero view override (YTD / Periodic)
    - Force balance accounts to YTD view (True / False)

      See *Integration Settings* under *Workflow Profiles* in *Workflow* for descriptions of these settings.
3. Analyze prior data loads
    - Evaluate previously loaded data units to list data units to clear during the load.
4. Execute clear data
    - User-defined workflow channel configuration not used
    - Clear Workflow data units loaded by the workflow unit. A workflow data unit considers accounts and cube data unit standard members, so data clears at an account level by default.

User-Defined Workflow Channel Configuration Execution Steps

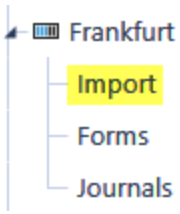
Clear all workflow channel data units loaded by the workflow unit. User-defined members and workflow data unit members are standard members of a workflow channel data unit, so data clears at a user defined member entity, scenario, time, and account level by default.

1. Execute load data
    - Data loads using parallel processing by entity. Multiple entities process at the same time.

Workflow Profile Data Loading Behaviors

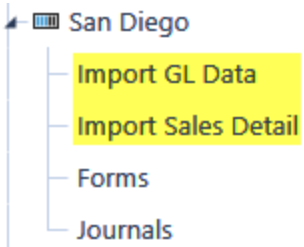
This section describes three specific data loading behavior patterns. These behaviors range from the basic data load process where one workflow profile loads one data unit, to more than one workflow profile loads one data unit.

Behavior 1: Single Workflow Profile - Loading One or More Entities on a Mutually Exclusive Basis:



This workflow profile configuration has only one import child profile under the parent (Frankfurt). The workflow engine follows basic clear and replace data loading steps described in Data Load Execution Steps (Clear and Replace).

Behavior 2: Multiple Import Child Workflow Profiles



This workflow profile configuration has more than one import child profile under the parent (San Diego). The workflow engine must perform extra steps to determine how to load data in the child profiles.

In this case the two import child profiles may try to load the same cube or workflow data unit because they have the same input parent workflow profile and are trying to load the same entities. When the import GL Data or Import Sales Detail workflow profiles execute the data load step, the following process determines how to correctly load data from both profiles to the cube.

Multiple Import Child Data Load Evaluation Steps:

1. Check for overlapped data units between import child siblings (Import GL Data or Import Sales Detail).
2. Determine the siblings have overlapped data units.

   If yes, clear all previously loaded data units for both import child siblings, then reload both using an accumulate load method in the order they appear in the workflow hierarchy. If the two siblings are loading to the same cells, the values are be added together and placed in the cell.

   If no, use the basic clear and replace data loading steps described in Data Load Execution Steps (Clear and Replace).

Behavior 3: Multiple Import Parent Workflow Profiles Loading One or More Common Entities



This workflow profile configuration has a central input parent profile that may load data assigned to another workflow profile. The Central HR Load workflow profile must have the Can Load Unrelated Entities set to **True**, so the workflow engine will let it try to write data for unassigned entities.

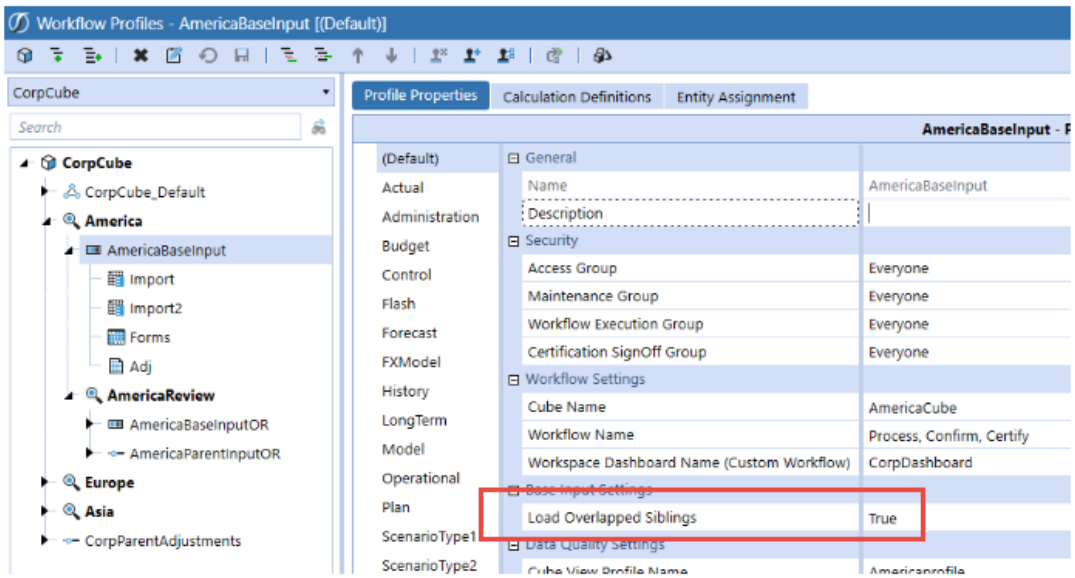
In this situation, when either Central HR Load or Houston executes a data load, the basic clear and replace data loading steps described in the previous section are used. However, Central HR Load does not control any entities so it checks and abides by the workflow and locking status of the workflow profiles that own the entities. For example, the workflow engine disallow updates if Houston is certified or locked and Central HR Load tries to load an entity owned by Houston.

Managing Sibling Imports

Certain application designs may require a workflow parent to have multiple sibling import channels. These designs typically use parallel processing techniques to load multiple non-overlapping sibling import children. The Load Overlapped Siblings setting on the parent boosts parallel processing performance in these workflow designs by eliminating overlapping checks between sibling channels. It only happens when the sibling channels' data sources do not contain overlapping data unit data records. This switch lets applications optimize data partitioning with parallel processing using the least workflow profiles.

Load Overlapped Siblings

- **True**: Default behavior, sibling channels check for overlapping data units.
- **False**: Do not check sibling channels for overlapping data units. If an overlapping condition occurs, the last processed channel overwrites the prior.



Data Locking

OneStream uses a locking strategy different than other analytic systems. All data control tasks are delegated to the workflow engine including the entity data locking control because of the integrated workflow controller.

The workflow engine creates a bidirectional link between the workflow engine, the staging engine, and the analytic engine. This two-way link creates a much stronger control structure compared to systems with separate workflow control modules that only interact with an analytic model in a unidirectional control structure.

This is an important control feature because if a user of the system attempts to update a data cell directly after all workflow processing is completed, the analytic engine must check with the workflow engine to determine if the cell can be updated. In a unidirectional control structure, the data cell can be unlocked and updated regardless of the workflow control state creating a break in the process audit chain. This situation cannot exist because every input data cell is associated with a workflow unit. Any attempt to update a data cell directly from a data entry form or Excel, for example, triggers the workflow engine to validate the data cell's workflow state by resolving its workflow status through the entity assignment relationship mechanism.

Locking data in means the data is locked for input. When data is locked (explicitly or implicitly) the workflow engine will not allow any form of data input to affect the entities assigned to the workflow profile of the locked workflow unit.

Lock Types

Explicit Locks: An explicit entity data lock is created when a workflow unit is locked therefore locking its assigned entities for the scenario and time associated with the workflow unit.

Implicit Locks

An implicit entity data lock is created when a workflow unit's parent workflow has been certified. Implicit locks are created to ensure after a higher-level workflow unit is certified, the underlying entity data cannot be changed. Implicit locks can be cleared by un-certifying the parent workflow unit.

Workflow Only Locks

If a workflow unit is locked and the workflow profile does not have assigned entities, all workflow processing is blocked, but there are no entity locks placed.

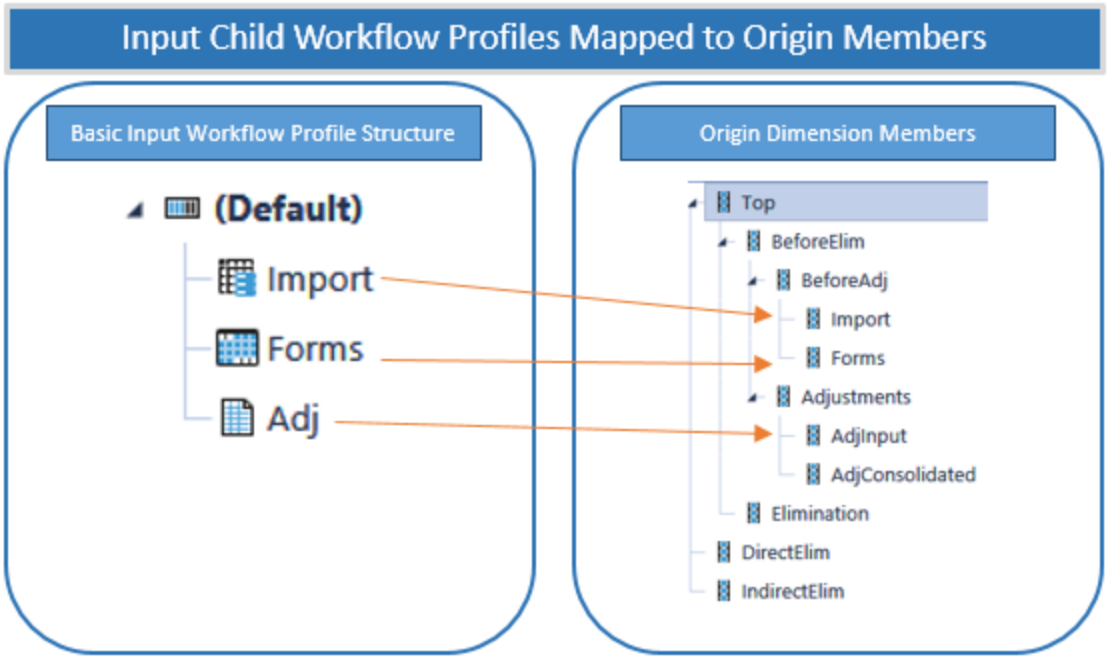
Locking Granularity

Data locks can be placed at different levels of granularity within the analytic model.

Input Parent Workflow Profiles (Level One Data Unit Lock): The highest level of locking occurs when an input parent workflow profile is locked. This is a level one data unit lock because the workflow engine will force all data cells for the entities assigned to the input parent to be locked. This is accomplished by locking the input parent workflow profile as well as the input child workflow profiles regardless of origin member binding or workflow channel assignment.

Input Child Workflow Profile (Origin Lock)

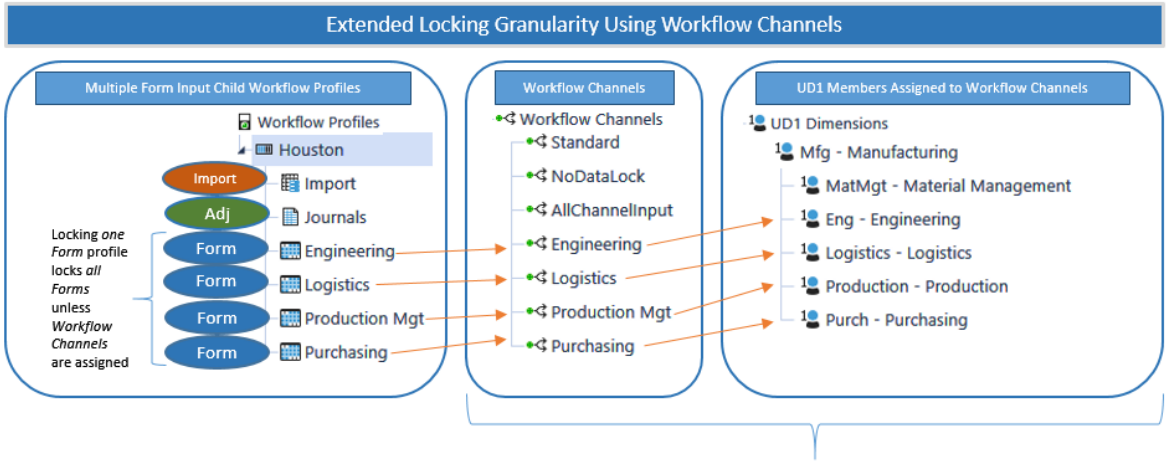
OneStream utilizes a predefined relationship between the three types of input child workflow profiles and a static dimension called the origin dimension to control and lock the three basic channels of input for an entity. This relationship provides the ability to control and lock data entry processes and associated data cells by binding the Workflow Profile Input Child Types to Origin Dimension Members.



Input Child Workflow Profile (Workflow Channel Lock)

Through the use of workflow channels assigned to input child workflow profiles and associated with accounts or user defined members, it is possible to provide very detailed locking granularity within a given origin member.

The diagram below demonstrates how the forms origin member has been divided into multiple workflow channels, enabling each form input child workflow profile and the UD1 data cells bound to the same workflow channel to be locked independently. Workflow channels can be used with Import Input Children as well Adjustment Input Children.



Assigning a different Workflow Channel to each Form Input Child and then to each corresponding UD1 Member increases the locking granularity to the level of the assigned Workflow Channel.

Example:
Locking the *Purchasing* Form Input Child will lock the intersections for the Entities assigned to Houston, where the Origin Member is Form and the UD1Member is Purch.

Security

To lock a workflow profile, you need workflow execution group membership or an inherited workflow execution group membership from an ancestor workflow profile.

The execution group authorization to lock workflows applies to all descendent workflow profiles regardless of the access group or workflow execution group member rights assigned to the descendants.

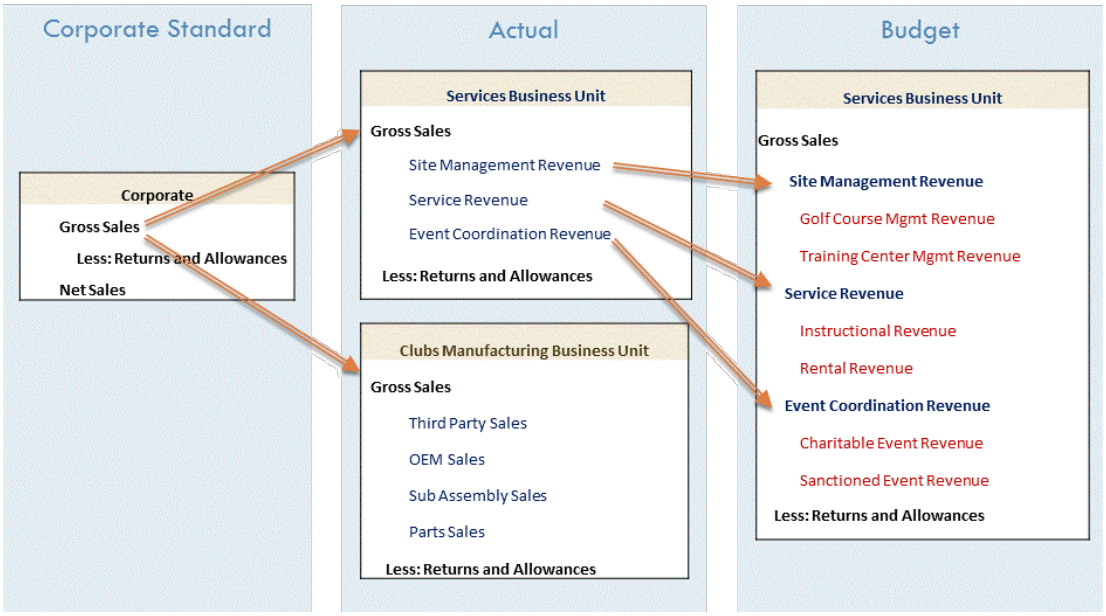
The audit workflow process provides an audit trail of user activity in the locking and unlocking workflow profiles

To unlock workflows as a non-administrator, you must have membership in the workflow execution group and the Application Security role UnlockWorkflowUnit. Unlocking descendent workflow profiles is inherited from workflow execution group membership on an ancestor workflow profiles

Extensible Dimensionality

Business units can inherit standard dimensions from a standard set that corporate may maintain, but Extensible Dimensionality® also allows them to extend those dimensions to suit their own process and reporting needs. This allows for operational significance for the business units yet grants control of the overall process to corporate.

The diagram below shows an example of how a certain account can be extended differently across a service business unit as opposed to a manufacturing business unit across the Actual and Budget scenarios. Notice how each business unit can look at gross sales differently in the Actual scenario. The services business unit can also look at gross sales at an even greater level of detail in their Budget scenario.



This is possible due to three reasons:

1. Extensible Dimensionality® can inherit dimensions and extend them. There are four different Account dimensions in the example that inherit from each other like this:

    - Corporate Accounts
    - Club Manufacturing Accounts
    - Services Accounts
    - Services Budget Accounts

    Corporate Accounts is the main chart of accounts. Club manufacturing takes that dimension and extends it to add its own accounts, but it cannot change what is in Corporate Accounts. Services also takes Corporate Accounts and extends it to meet its needs for the Actual scenario and extends its own Services Accounts to meet its need for more detail in its budget scenario in the Services Budget Accounts dimension. However, when designing an extended dimension, members are either added below inherited members for additional detail, or to an alternate roll-up by creating new parent members and then referencing inherited members below the new parents. Both actions cannot be done at the same time in the same section of the hierarchy.

2. Different dimensions can be assigned to different cubes and that dimensional assignment can be different for each Scenario Type. In the example, there are three cubes: Corporate, Clubs, and Services. When looking at the Corporate cube, data from the three cubes is there for analysis. In Corporate, the Corporate Accounts dimension is assigned to all scenario types. In Clubs, the Clubs Manufacturing Accounts dimension is assigned to all scenario types. In the Services cube, the Services Accounts dimension is assigned to every dimension except for Budget, which is where the Services Budget Accounts dimension is assigned.

3. The Clubs and Services cubes have their own respective entity dimensions referenced in the Corporate cube. The entity dimensions tie the data together.

> **NOTE:** Other dimensions such as flow and the user defined dimensions can also be extended and have flexible cube assignment if needed.

> **IMPORTANT:** Using Extensible Dimensionality® to extend accounts used in the Intercompany Matching process is not a recommended practice.

Extensibility concepts are broken down into the following categories:

- Vertical Extensibility
- Workflow Extensibility
- Platform Extensibility
- Horizontal Extensibility

Vertical Extensibility

Vertical or Entity/Cube extensibility relates to the ability to include or exclude detail at different levels up the entity hierarchy. It is important to properly manage data unit size to enable optimal performance while accounting for future growth. Vertical extensibility also relates to varying dimensionality across business units. With vertical extensibility, lower-level entities still report at a detailed level, but the data can collapse to a summary level to facilitate reporting and increase performance. Vertical extensibility provides flexibility to vary dimensionality at a detailed level but consolidate to a common summary level. There can also be separate cubes for different uses, such as human resource data or cost drivers. These cubes can still reference other cube data through the CB# qualifier in Member formulas.

Workflow Extensibility

Workflow extensibility enables variation in the input steps and methods within each process flow. Workflow steps and settings are adjustable in each scenario type, or can be combined if multiple processes follow the same responsibility hierarchy. You can configure workflow extensibility on each parent cube to tailor the software interface to match process needs. There can be different Workflow Profile hierarchies per scenario type which is defined at the cube level.

> **Example:** An actual scenario might be loaded from 12 GL systems across 500 entities, and Budget Forecast and Variance can define a Workflow for each of the 500 entities with regional review and sign off levels.

If your actual data collection process is more import driven, and the planning process driven by calculations and dashboards, workflow extensibility can split these processes to ease management from an administrative standpoint. Some data collections may require centralized import while others are distributed to more end users. You can assign entities only once in a workflow hierarchy, so workflow extensibility can be used to vary entity sign off responsibilities and allow for differing entity assignments.

Platform Extensibility

Platform extensibility relates to variation in data storage locations and how it is utilized within the platform. Platform extensibility also enables multiple applications within one environment that can talk to each other. OneStream has the unique ability to consume, utilize, and report on data regardless of its storage in cubes, relational tables, or even external locations. With platform extensibility, cube data can combine with relational data to achieve optimal balance between performance and reporting needs. Platform extensibility also allows use of relational data, web content, and external data sets.

Horizontal Extensibility

Horizontal Extensibility relates to the extension and use of different levels in a hierarchy for different business purposes. It also allows targeting of when and where the data model needs to include dimensions.

> **Example:** You can input data at a parent level. A parent becomes a base for input in a different scenario by using the scenario type settings and properly applying cube dimension assignments. See Cube Dimensions.

> **Example:** If you have highly detailed metadata that only applies in a specific use case, horizontal extensibility can help limit potential intersections that are invalid for all other use cases by assigning metadata only where it is relevant.

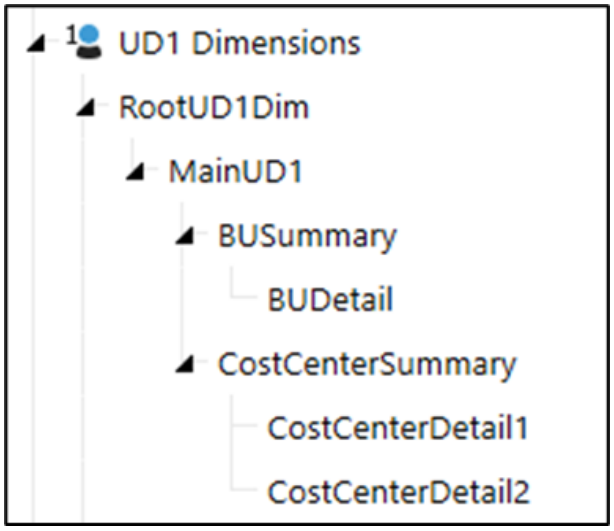
Recommended Implementation of Extensibility

There is not a universal way to implement extensibility in OneStream. Applications without extensibility commonly run into issues. A lack of vertical and platform extensibility can lead to performance issues. A lack of horizontal and workflow extensibility can lead to flexibility issues. Applications with too much extensibility encounter fewer performance or flexibility issues, but do have a higher maintenance burden. Balancing performance, usability, and maintenance is key. In most cases, it is recommenced that extensibility is considered in every design and implemented . When designing a solution, building a matrix can help you visualize the details and where they need inclusion, and then you can start shaping scenario types, cubes, dimensions, and any platform extensibility.

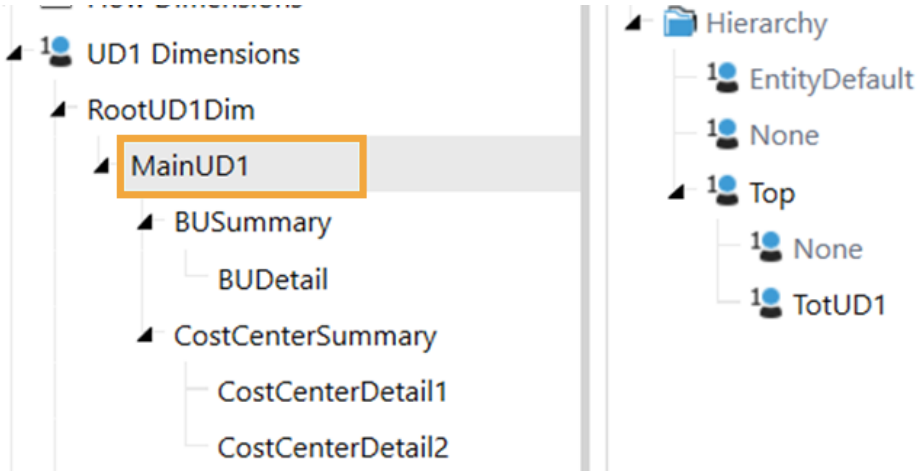
Cube & Scenario Dimension Assignment Matrix

Scenario:	Actual				Budget				Forecast				Relational Tables			
Cubes:	SUMMARY	CUBE 1	CUBE 2	CUBE 3	SUMMARY	CUBE 1	CUBE 2	CUBE 3	SUMMARY	CUBE 1	CUBE 2	CUBE 3	EMPLOYEE	ACTUALS	ASSETS	VENDORS
Entity																
SummaryMgmt	X				X				X							
Division 1 Entities		X				X				X						
Division 2 Entities			X				X				X					
Division 3 Entities				X				X				X				
Account																
RootAccountDim																
Summary Accounts	X				X				X							
Forecast Accounts										X	X	X				
Budget Accounts						X	X	X								
Actual Accounts		X	X	X												
Flow																
RootFlowDim																
MainFlow	X	X	X	X	X	X	X	X	X	X	X	X				
UD1: Program																
RootUD1Dim				X				X				X				
MainUD1																
Summary Program	X				X				X							
Division 1 Plan Programs						X				X						
Division 1 Actual Programs		X														
Division 2 Programs			X				X				X					
UD2: Center																
RootUD2Dim																
MainUD2																
Summary Centers	X				X	X	X	X	X	X	X	X				
Detail Centers		X	X	X												
UD3: Region																
RootUD3Dim																
MainUD3																
Region	X				X				X	X	X	X				
Country			X	X		X	X	X								
Location		X														

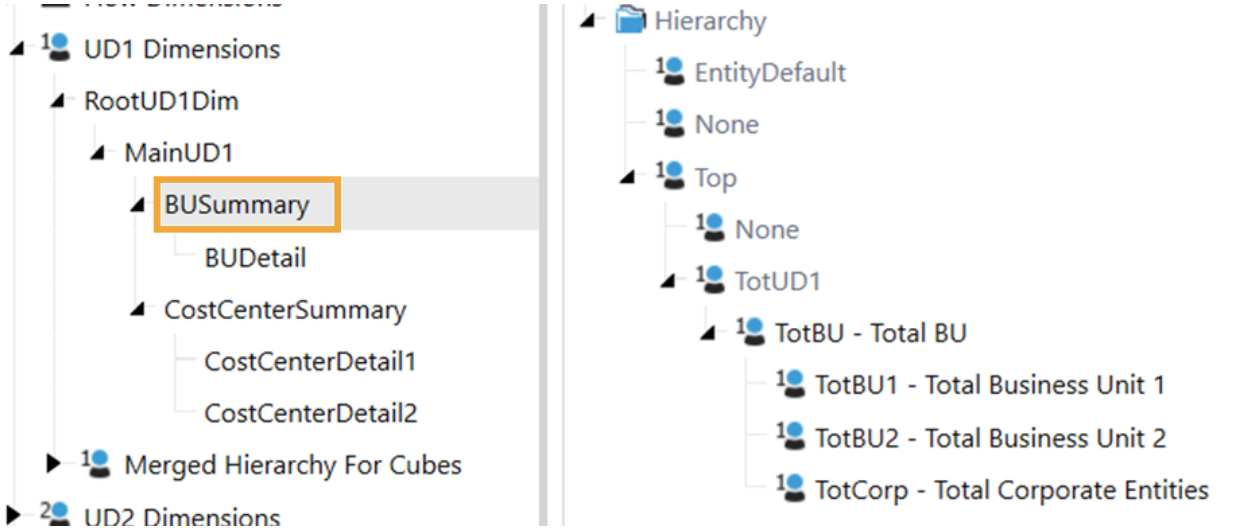
The CPM Blueprint application contains examples of extensibility configurations using general best practices. UD1 is an example of using the common configuration of a MainUD1 dimension parent to summarize the BU and Cost Center details into a common dimension.



Applying this concept to all user defined dimensions facilitates both vertical and horizontal extensibility. To create vertical extensibility, dimensional detail that is not needed in a parent cube can be collapsed by assigning MainUD1.



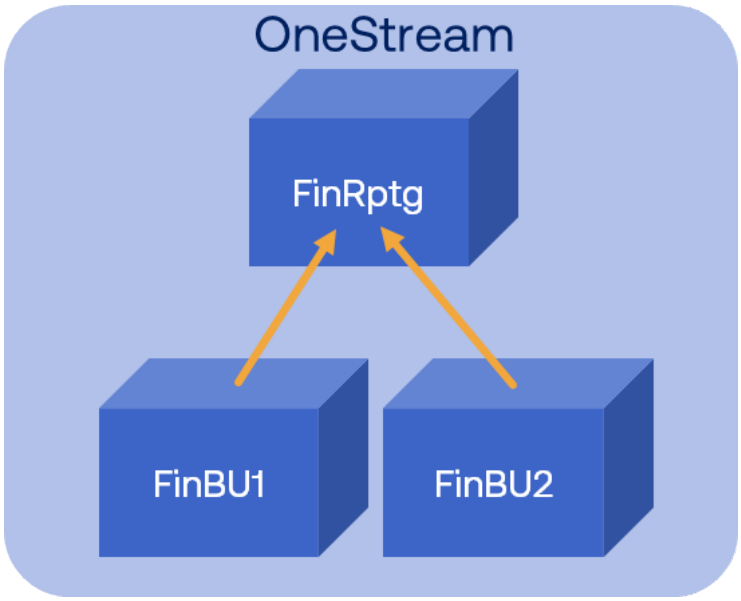
The dimensional detail is the extended from TotUD1 to expand into necessary levels of detail for each data collection and reporting need. This allows both None and Top to be active at all levels in the dimensional hierarchy.



Another example of extensibility in the CPM Blueprint application is in the cube configuration. The configuration focuses on the financial reporting structure in this application and follows the recommendation for a base-summary cube relationship between business unit and total company reporting. This configuration has several benefits:

- More flexibility to expand child cubes horizontally and plug in different dimensionalities
- Greater ability to collapse the data unit if its size becomes an issue

- Further future-proofing by enabling more platform expansion with the same foundation



This application also has workflow extensibility on display. You can see the connection between top level and base level cubes in the cube settings, and the workflow suffixing applied in the CPM Blueprint application. In this example, the Actual scenario type has a different process flow and responsibility hierarchy from other data collections, so it has own suffix of ACT Budget and Forecast follow the same process flow and responsibility hierarchy, so they share a workflow suffix of BUDFCST This enables each process to have its own configuration and entity assignment.

Cube Properties | Cube Dimensions | Cube References | Data Access | Integration

⊟ General	
Name	FinRptg
Description	Financial Reporting
Cube Type	Standard
Time Dimension Profile	Standard
⊟ Security	
Access Group	Everyone
Maintenance Group	Everyone
Use Parent Security for Relationship Consolidation	False
⊟ Workflow	
Is Top Level Cube For Workflow	True
Suffix for varying Workflow by Scenario Type	
Actual	ACT
Administration	
Budget	BUDFCST
Control	
Flash	
Forecast	BUDFCST

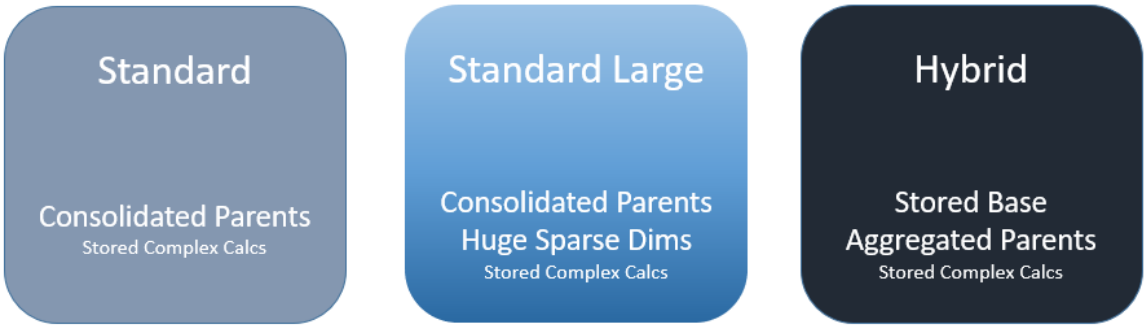
Cube Design Choices within an Application

There are a few choices for the use of cubes within an application, driven by the relationships between data.

1. Single or **Monolithic Cube** is the simplest application design. These typically have one Workflow Profile structure, though that can vary by Scenario type.
2. **Linked Cubes** is possible via relationships between multiple Entity dimensions into one superset Entity dimension. The top or parent cube is configured with cube references to others. Typically, extensible dimensionality® is deployed with other dimensions, such as Accounts, allowing the business unit cubes to satisfy their management reporting requirements. There is typically one Workflow Profile structure for all linked cubes, though that can vary by Scenario type.
3. **Exclusive Cubes** are separate business unit cubes that move their data from a business unit to a parent cube typically via business rules or the use of data management instead of through configured cube references. Each separate cube requires its own Workflow Profile structure, though that can vary by Scenario Type.
4. **Specialty Cubes** refers to special data collections outside of the typical trial balance or planning data loads and is typically encompassed with no parent/child relationships between cubes. Examples are for headcount or budget drivers. These figures might be referenced by other cubes via business rules or data management. Each specialty cube would have separate Workflow Profile structures.

Application Model Design Considerations

Application designers understand that, as a platform, each application must flexibly respond to growth, evolving business needs, and enhancements in OneStream Solutions. Review the following design options to best direct the overall design and performance expectations of your application.



When designing an Application, consider these questions to identify the model cube design best suited to your needs:

- What is the overall size of the Entity, Account, Flow and User Defined dimensions that estimate the Data Unit size potential? A OneStream Data Unit is comprised of Cube, Entity, Parent, Consolidation, Scenario and Time.

- What is the size and relationship between dimensions? For example, are the User Defined Dimensions large (thousand members) and are the members closely tied to a single or very few Entities?
- Are dimension members more transactional in nature? Transaction members such as Employee ID, SKU and Project Code likely change frequently. While this level of detail may be needed for analysis, it can complicate a financial model. Other modeling techniques could help.
- Is the data related to the cube a “specialty” collection, such as Human Resources, Product Profit or Capital Expenditures?
- Do you fully understand how the data is managed? Determine if the Legal Entity is appropriate for the Entity dimension so you can properly define the ultimate Data Unit Controlling Dimension.
- Have you collected information how the data is assembled and used? Determine if the Cube data is integrated for different types of data such as Actual and Budget.

Standard Cube Design

Following are considerations for standard cube design:

Guidelines

- Common design for a standard Statutory Consolidation Cube
- Primary metadata driven application
- Data Unit approximately 250k data records at higher end

Performance Profile

- Consolidation times for 12 periods generally range from minutes to hours on the higher end, depending on complexity and volume
- Performance generally impacted by metadata parent structures, alternate hierarchies, and calculations executing on parent Entity members

Reporting Guidelines

- Standard Cube Views with minimum number of rows and columns should perform well, rendering within a few seconds to a minute (depending on requirements)
- Slow Cube View performance of a reasonable number of non-calculated rows and columns, rendering greater than a minute should flag a Cube View or Application design review
- Cube View Performance is strongly dependent on concurrent use, Cube View design and metadata involved

Standard Large Sparse Cube Design

Following are considerations for standard large sparse cube design:

Design Guidelines

- Common for a Statutory Consolidation Application. Contain sparse intersections due to collecting transactional members, such as SKU, Part Number or Names, or other large dimensions, such as Product or Project

- Data Unit size approximately 750k or more data records. Larger Data Units demand more CPU processing power and time to calculate

Performance Profile

- Optimize consolidation times by addressing sparse dimensions. With good extensibility you can significantly reduce the size of the Data Unit consolidating, for faster consolidation
- Applications that must be consolidated and contain large sparse dimension members, may have long consolidation times and lower overall performance. Review these applications designs to consider alternate design solutions that include extensibility or other cube and Model solutions.
- Data Unit size, number of Data Units, Business Rule best practices and other factors effect consolidation times

Reporting Guidelines

- Putting a “top level” member such as UD2#Top for a large dimension on a Cube View , means OneStream loops through each child member and aggregates on-the-fly. This can consume large amounts of processing power and extend report processing.
- It is best to use Sparse Data Suppression settings on Cube Views which are based on Single Data Unit.
- You may need custom Sparse Cube View Business Rules for “top level” queries.
- Quick Views may be impacted as they do not include Sparse Row Suppression settings.

Hybrid Cube Design

The following are considerations for hybrid cube design.

Guidelines

Hybrid Applications are generally designed to support Analytics and not the Consolidation model. Hybrid cubes:

- Typically require complex calculations at base level Entities.
- Predominately present in aggregation having simple needs at parent Entity levels. Parents do not require special calculations or logic.
- Rely on Dynamic Business Rules attached to a dynamic member such as in UD8. This means parent Entity data is not stored. You can set the Is Consolidated property on these Entities to False.
- Determine if parent Entity data must be exported to external system so you can understand the impact of Dynamic Business Rules.
- Usually have a data unit size of up to one million data records at the top parent Entity level.

Performance Profile

- Performance will vary by application size and structure.
- Data calculation is not stored at the parent level.
- “Entity Aggregation on-the-fly” consolidation simulation rules are required. You cannot consolidate parent entities. This involves the API.Functions. GetEntityAggregationDataCell function in a dynamic member.
- You must support the export of calculated data which could involve unique Scenarios or cubes to replicate dynamic calculations as stored values. These processes can be automated using scheduled Data Management jobs.

Reporting Guidelines

- Apply Sparse Suppression settings on Cube Views.
- You may need custom Sparse Cube View Business Rules for “top level” member queries.
- Intermediate rules may run as Workflow-based or be driven by Dashboards or Calculation Definition settings.
- Quick Views may be impacted because they do not include Sparse Row Suppression settings.

Relational Blending

The dynamic blending of data stored in a relational database with data stored/derived by the in-memory analytic engine is supported.

Relational blending is a data management approach. This enables analytic modelers to better manage the trade-offs between building unsustainable/poor performing complex analytic models that contain too much detail and the need to report on that level detail (data that may be transactional in nature or constantly changing). Relational blending makes it possible to seamlessly integrate detail data points into the analytic reporting process.

Relational Blending Types

There are three methods of relational blending.

1. **Drill-Back Blending (One-to-Many Relationship)**
   This method of relational blending is used to provide access to detailed information that does not exist in the analytic model. This capability is delivered right out-of-the box with its predefined drill back to stage detail data. In addition, the stage integration engine provides drill-back and drill-around capabilities against external data.

2. **Application Blending (One-to-Many Relationship)**
   This method of relational blending leverages theOneStream Solution specialty planning and compliance applications. This collects information in a transactional register format and seamlessly maps/loads summarized data into an analytic model. These applications also provide predefined transactional level reports as well as predefined drill-back connectors allowing drill-down from a summarized analytic model to the detailed register transaction data.
3. **Model Blending (One-to-One Relationship)**
   This method of relational blending combines the power of the in-memory analytic engine with the flexibility of relational database storage. This functionality is provided as part of the Finance Engine API and the functions can be found under the api.Functions path in a finance business rule.

Model Blending

The following sections provide use-cases and explanations of how model blending is used to seamlessly integrate relational data points into an analytic model using dynamic member calculations that leverage the relational blending API functions of the finance engine.

A common challenge in analytic modeling is how to build a sustainable model when the definition of metadata and data becomes blurred. The two use cases below provide examples of this modeling challenge.

Use Case 1

In many cases, analytic modelers are faced with the challenge of building a model containing dimension members that are unknown at design time or forced to build a dimension containing members that will be constantly changing. Without relational blending, analytic modelers are forced to build models full of unknown members (TBD1, TBD2, etc.) with the hopes that users of the system do not need values beyond these placeholder members. (This data is transactional and not a good candidate for an analytic model; Workforce Planning is a good example of this problem.)

This is a one-to-many issue so drill-back and application blending work well if the summary cube is the primary focus of analysis and transactions are only used for supporting details. Model blending can provide a benefit as well, but keep in mind that model blending must relate an analytic cell POV to a relational row or summarized row (one-to-one). For Model blending to be useful in these circumstances, the relational data must be returned in an aggregate format (avg, min, max, sum, count) in order to reduce the one-to-many relationship to a one-to-one relationship.

Use Case 2

Analytic models that depend on dimensions with members that are constantly changing. Consider business problem where the analytic model is based on a fixed number of members (facility with rooms and beds). This is easy from a modeling perspective; however, the user requirement is to build a model that is aware of the current occupant of the bed. The logical metadata definition is room/bed, but the business problem requires the “occupant” to be defined as a member for the model to be meaningful. If occupant is used as a member in the model, it is almost guaranteed that the analytic model will eventually become unsustainable due to the changing nature of the room/bed/occupant dimension. The administrator of this model now has the burden of constantly changing and rebuilding the model to reflect the current occupant data.

This is a one-to-one issue, so model blending fits well and provides a tremendous amount of value. Detailed and changing information can be continuously loaded and updated as attribute information in the OneStream staging tables, custom relational tables and the Model Blending API can be used to dynamically incorporate this information into analytic model through dynamic member formulas.

Model Blending Benefits

Relational blending is similar to the OneStream staging engine in that it is a tool to protect the analytic engine. Analytic modelers are aware that there is a powerful force with which they must contend when they are trying to create a well-designed, well-performing and maintainable model. That powerful force is factorial combination math. Analytic modelers understand that numbers of possible cells in a model (combinations) is determined by the number of members in each dimension multiplied by each other (1,000 accounts x 100 cost centers x 10,000 employees x 20 regions = 20 billion combinations). This phenomenon means that analytic modelers are in a constant battle. They are trying to capture the data points required to understand the business process being modeled and model performance challenges created by the computational physics of factorial combination math. In summary, it is easy to create an analytic model with a massive number of potential cells and as a result, end up with a poor performing model.

Relational blending can help keep the size of an analytic model to a manageable level by allowing leaf level members to be kept in the relational table and only keeping summarize/static Members in the analytic model definition (dimension members). Relational blending is not a cure-all, but it is an important tool for building maintainable well performing analytic models when a model has some dependencies on detailed information that cannot be clearly defined as metadata or data. In other words, the information is useful in the model, but it is so detailed or changes so much that it is difficult to incorporate into a rational metadata structure.

Relational Blending API

The Relational Blending API functions listed below can be used to efficiently lookup a value in a cached relational table. This is based on the current POV values of the analytic engine or by providing specific override values for the cube name, entity name, scenario name, time name and account name. It is important to understand how caching of the data table is done. The CacheLevel parameter is used to control the cache granularity which will in turn control cache efficiency. To choose the most efficient CacheLevel value, determine how the data will be used in a cube view. If the primary Cube View data request will be on a single data unit (entity, scenario, time), then the best cache level choice would be BlendCacheLevelTypes.WfProfileScenarioTimeEntity.

This is an efficient choice because the first time a cell is requested for the entity, scenario, time combination, a query will run to load all the stage data for the combination. Then all subsequent cell requests would read values from cache. On the other hand, if the primary Cube View data request is for multiple entities, then BlendCacheLevelTypes.WfProfileScenarioTime cache level would be a more efficient choice. This is more efficient because a single query would run to load all the data for the scenario and time into cache and then all subsequent cell requests would read values from cache.

> **CAUTION:** Be aware that using coarse cache levels (reading more data at once into cache) only benefits performance when the target cube view can read many values for the cache. If the target cube view is only focused on one entity/account and the BlendCacheLevelTypes.WfProfileScenarioTime cache level is chosen, all rows for the entire scenario and time would need to be read into memory when only values for one entity/account combination was needed. In this case, BlendCacheLevelTypes.WfProfileScenarioTimeEntityAccount would be a more efficient cache level.

In summary, choose a cache level that will minimize the number of actual database queries needed to run in order to get the desired cells for the target Cube vVew. This is not an exact science, and it may be difficult to choose a cache level that works efficiently for all target Cube Views. If there is a diverse set of Cube Views using relational blend data, consider creating specific members that implement different cache levels that match the Cube View data pattern.

Cache Level Types

BlendCacheLevelTypes.WfProfileScenarioTime
Query will be run and cached using the supplied workflow profile, POV scenario and POV time as criteria and cache key.

BlendCacheLevelTypes.WfProfileScenarioTimeEntity
Query will be run and cached using the supplied workflow profile, POV scenario, POV time and POV entity as criteria and cache key.

BlendCacheLevelTypes.WfProfileScenarioTimeAccount
Query will be run and cached using the supplied workflow profile, POV scenario, POV time and POV account as criteria and cache key.

BlendCacheLevelTypes.WfProfileScenarioTimeEntityAccount
Query will be run and cached using the supplied workflow profile, POV scenario, POV time, POV entity and POV account as criteria and cache key.

BlendCacheLevelTypes.Custom
Intended to be used with custom table query (cache level is explicitly controlled by the supplied SQL query). Query will be run and cached using the supplied cache name.

Relational Model Blending API Functions

GetStageBlendTextUsingCurrentPOV

Function Prototype

Public Function GetStageBlendTextUsingCurrentPOV (ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal wfProfileName As String, ByVal fieldList As String, ByVal criteria As String, ByVal fieldToReturn As String, ByVal textOperation As BlendTextOperationTypes) As String

Description

Read a stage text attribute value from a cached ado.net data table using the current POV values and optionally perform concatenation on the results.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cacheLevel

Cache granularity level used to control how much information is cached in each chunk (less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary).

cacheName

Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

wfProfileName

Name of the import workflow profile containing the values to be looked up. (Pass an empty string to look up the workflow based on the POV entity, use *.YourWFSuffix to get workflow profiles with the specified suffix.)

fieldList

List of STAGE fields that will be used as criteria and/or returned.

Criteria

Criteria statement used to select rows in the cached data table.

fieldToReturn

Name of the stage field to return.

textOperation

Text operation to perform on the resulting data table

> **NOTE:** FirstValue returns the first matching row if there is more than one stage value for the specified cell.

Return Type

String

Example:

```
'UD8 DynamicCalc - Lookup Attribute 1 From Stage
If Not api.Entity.HasChildren Then
Dim criteria As New Text.StringBuilder
criteria.Append("U1T = '" & api.Pov.UD1.Name & "' ")
criteria.Append("And U2T = '" & api.Pov.UD2.Name & "' ")
Return api.Functions.GetStageBlendTextUsingCurrentPov(BlendCacheLevelTypes.
        WfProfileScenarioTimeEntity, "DU",
        "*.Sales Detail", "U1T,U2T,A1,ConvertedAmount", criteria.ToString,
            "A1", BlendTextOperationTypes.FirstValue)

Else
Return String.Empty
End If
```

GetStageBlendText

Function Prototype

Public Function GetStageBlendText (ByVal cubeName As String, ByVal entityName As String, ByVal scenarioName As String, ByVal timeName As String, ByVal accountName As String, ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal wfProfileName As String, ByVal fieldList As String, ByVal criteria As String, ByVal fieldToReturn As String, ByVal textOperation As BlendTextOperationTypes) As String

Description

Read a stage text attribute value from a cached ado.net data table using the specified POV values and optionally perform concatenation on the results.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cubeName

Name of the cube to use for the POV.

entityName

Name of the entity to use for the POV.

scenarioName

Name of the scenario to use for the POV.

timeName

Name of the time to use for the POV.

accountName

Name of the account to use for the POV.

cacheLevel
Cache granularity level used to control how much information is cached in each chunk (less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary).

cacheName
Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

wfProfileName
Name of the import workflow profile containing the values to be looked up (pass an empty string if to look up the workflow based on the POV entity, use *.YourWFSuffix to get workflow profiles with the specified suffix).

fieldList
List of stage fields that will be used as criteria and/or returned.

Criteria
Criteria statement used to select rows in the cached data table.

fieldToReturn
Name of the stage field to return.

textOperation
Text operation to perform on the resulting data table

> **NOTE:** FirstValue returns the first matching row if there is more than one stage value for the specified cell.

Return Type
String

GetStageBlendNumberUsingCurrentPOV

Function Prototype

Public Function GetStageBlendNumberUsingCurrentPOV(ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal wfProfileName As String, ByVal fieldList As String, ByVal criteria As String, ByVal fieldToReturn As String, ByVal mathOperation As BlendNumericOperationTypes) As Decimal

Description

Read a stage numeric attribute value from a cached ado.net data table using the current POV values and optionally perform aggregation math on the results.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cacheLevel

Cache granularity level used to control how much information is cached in each chunk (less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary).

cacheName

Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

wfProfileName

Name of the import workflow profile containing the values to be looked up (pass an empty string if you want to look up the workflow based on the POV entity, use *.YourWFSuffix to get workflow profiles with the specified suffix).

fieldList

List of stage fields that will be used as criteria and/or returned.

criteria

Criteria statement used to select rows in the cached data table.

fieldToReturn

Name of the stage field to perform math on and return.

mathOperation

Math operation to perform on the resulting data table

> **NOTE:** FirstValue returns the first matching row if there is more than one stage value for the specified cell.

Return Type

Decimal

Example:

```
'UD8 DynamicCalc - Lookup Average ConvertedAmount From Stage
If Not api.Entity.HasChildren Then
    Dim criteria As New Text.StringBuilder
    criteria.Append("U1T = '" & api.Pov.UD1.Name & "' ")
    criteria.Append("And U2T = '" & api.Pov.UD2.Name & "' ")
    Return api.Functions.GetStageBlendNumberUsingCurrentPov(BlendCacheLevelTypes.
    WfProfileScenarioTimeEntity, "DU",
    "*.Sales Detail", "U1T,U2T,A1,ConvertedAmount", criteria.ToString,
        "ConvertedAmount",
    BlendNumericOperationTypes.AverageSkipZero)
Else
    Return 0
End If
```

GetStageBlendNumber

Function Prototype

Public Function GetStageBlendNumber(ByVal cubeName As String, ByVal entityName As String, ByVal scenarioName As String, ByVal timeName As String, ByVal accountName As String, ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal wfProfileName As String, ByVal fieldList As String, ByVal criteria As String, ByVal fieldToReturn As String, ByVal mathOperation As BlendNumericOperationTypes) As Decimal

Description

Read a stage numeric attribute value from a cached ado.net data table using the specified POV values and optionally perform aggregation math on the results.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cubeName

Name of the cube to use for the POV.

entityName

Name of the entity to use for the POV.

scenarioName

Name of the scenario to use for the POV.

timeName

Name of the time to use for the POV.

accountName

Name of the account to use for the POV.

cacheLevel

Cache granularity level used to control how much information is cached in each chunk. Less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary.

cacheName

Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

wfProfileName

Name of the import workflow profile containing the values to be looked up. Pass an empty string to look up the workflow based on the POV entity, use *.YourWFSuffix to get workflow profiles with the specified suffix.

fieldList

List of stage fields that will be used as criteria and/or returned.

criteria

Criteria statement used to select rows in the cached data table.

fieldToReturn

Name of the stage field to perform math on and return.

mathOperation

Math operation to perform on the resulting data table

> **NOTE:** FirstValue returns the first matching row if there is more than one stage value for the specified cell

Return Type

Decimal

GetStageBlendDataTableUsingCurrentPOV

Function Prototype

Public Function GetStageBlendDataTableUsingCurrentPOV(ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal wfProfileName As String, ByVal fieldList As String) As DataTable

Description

Read stage data into a cached ado.net data table using the current POV values so that it can be Queried / Analyzed in memory on the application server.

NOTE: Cache only lives for the duration of the WCF call.

Parameters

cacheLevel

Cache granularity level used to control how much information is cached in each chunk (less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary).

cacheName

Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

wfProfileName

Name of the import workflow profile containing the values to be looked up (pass an empty etring to look up the workflow based on the POV entity, use *.YourWFSuffix to get workflow profiles with the specified suffix).

fieldList

List of stage fields that will be used as criteria and/or returned.

Return Type

DataTable

Example:

```
'Lookup Attribute 1 From Stage Cached Data Table
If Not api.Entity.HasChildren Then
        Dim result As String = String.Empty

'Get the DataTable from cache
Using dt As DataTable = api.Functions.GetStageBlendDataTableUsingCurrentPov
(BlendCacheLevelTypes.WfProfileScenarioTimeEntity, "DU", "*.Sales Detail",
"U1T,U2T,A1,ConvertedAmount")
    If Not dt Is Nothing Then
    'Execute a query against the data table and return the first matching row
    Dim criteria As New Text.StringBuilder
```

```
    criteria.Append("U1T = '" &amp;amp; api.Pov.UD1.Name &amp;amp; "' ")
    criteria.Append("And U2T = '" &amp;amp; api.Pov.UD2.Name &amp;amp; "' ")
    Dim rows As DataRow() = dt.Select(criteria.ToString)
    If rows.Count &gt; 0 Then
    result = rows(0)("A1")
        End If
      End If
  End Using
  Return result
Else
    Return String.Empty
End If
```

GetStageBlendDataTable

Function Prototype

Public Function GetStageBlendDataTable(ByVal cubeName As String, ByVal entityName As String, ByVal scenarioName As String, ByVal timeName As String, ByVal accountName As String, ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal wfProfileName As String, ByVal fieldList As String) As DataTable

Description

Read stage data into a cached ado.net data table using the specified POV values so that it can be Queried / Analyzed in memory on the application server.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cubeName

Name of the cube to use for the POV.

entityName

Name of the entity to use for the POV.

scenarioName

Name of the scenario to use for the POV.

timeName
Name of the time to use for the POV.

accountName
Name of the account to use for the POV.

cacheLevel
Cache granularity level used to control how much information is cached in each chunk. The less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary.

cacheName
Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

wfProfileName
Name of the import workflow profile containing the values to be looked up (pass an empty string to look up the workflow based on the POV Entity, use *.YourWFSuffix to get workflow profiles with the specified suffix).

fieldList
List of stage fields that will be used as criteria and/or returned.

Return Type
DataTable

GetCustomBlendDataTableUsingCurrentPOV

Function Prototype
Public Function GetCustomBlendDataTableUsingCurrentPOV(ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal sourceDBLocation As String, ByVal sourceSQL) As DataTable

Description

Read data from a custom table into a cached ado.net data table using the current POV values so that it can be Queried / Analyzed in memory on the application server.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cacheLevel

Cache granularity level used to control how much information is cached in each chunk (less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary).

cacheName

Short name used to identify the values placed in the cache (full CacheID will be CacheName + CacheLevel values).

sourceDBLocation

Database location name to query (application, framework, or a named external connection).

sourceSQL

SQL statement that defines the DataTable to cache for in memory querying.

Return Type

DataTable

Example:

```
'Lookup Invoice Number From a Custom Table
If Not api.Entity.HasChildren Then
        Dim result As String = String.Empty

        Dim queryToCache As New Text.StringBuilder
        queryToCache.Append("Select * ")
          queryToCache.Append("From InvoiceMaterialDetail ")
          queryToCache.Append("Where PlantCode = 'H200' And CustId = 'NH2421' And    InvYear
```

```
= 2011

        and InvMonth = 'M3'     ")
'Get the DataTable from cache
Using dt As DataTable =  api.Functions.GetCustomBlendDataTableUsingCurrentPov
(BlendCacheLevelTypes.
    Custom, "Material",
    "Revenue Mgmt System", QueryToCache.ToString)
    If Not dt Is Nothing Then
'Execute a query against the CUSTOM data table and return column InvNo for the first
matching  row
    Dim criteria As New Text.StringBuilder
     criteria.Append("WorkDay = 15")
    Dim rows As DataRow() = dt.Select(criteria.ToString)
    If rows.Count > 0 Then
    result = rows(0)("InvNo")
    End If
    End If
End Using
   Return result
Else
    Return String.Empty
End If
```

GetCustomBlendDataTable

Function Prototype

Public Function GetCustomBlendDataTable(ByVal cacheLevel As BlendCacheLevelTypes, ByVal cacheName As String, ByVal sourceDBLocation As String, ByVal As String) As DataTable

Description

Read data from a custom table into a cached ado.net data table using the specified POV values so that it can be Queried / Analyzed in memory on the application server.

> **NOTE:** Cache only lives for the duration of the WCF call.

Parameters

cacheLevel
Cache granularity level used to control how much information is cached in each chunk. The less granular cache level helps repeated calls but hurts requests for a single cell because more data is cached than necessary.

cacheName
Short name used to identify the values placed in the cache (Full CacheID will be CacheName + CacheLevel Values).

sourceDBLocation
Database location name to query (application, framework, or a named external connection).

sourceSQL
SQL statement that defines the DataTable to cache for in memory querying.

Return Type
DataTable

BI Blend

BI-Blend is a read-only aggregate storage model that supports large-volume reporting for data that is not appropriate to store in a traditional cube. BI Blend data is large in volume and often transactional. For example, to analyze data by invoice, a standard cube requires metadata to store the data records. Soon, most of the invoice metadata would not be needed given the transactional nature of the data. Therefore, storing transaction data in a cube design is not a best practice.

A key challenge of reporting on transactional data is presenting data in a uniform format for standardized reporting, while flexibility supporting ever changing records and reporting requirements. The overall large size of the data sets requires a model suitable for responsive reporting and analysis.

The BI Blend solution approaches these challenges in a unique and innovative way. BI Blend rationalizes source data for uniform and standardized reporting, like the standard cube models, but stores the data in a new relational column store table for responsive reporting.

Use BI Blend to analyze large volumes of highly changing data, such as ERP system transaction data, which typically would not reside in a cube. Processing is free 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

- Leveraged metadata, 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

For more information, see the *BI Blend Design and Reference Guide*.

Dimensions Overview

A dimension is like a folder used to organize business data in OneStream. It contains members that represent different types of data. For example, the Entity dimension contains members that represent the business units of an organization, and the account dimension members represent the account structure of an organization.

There are configurable and system-defined dimension types. Configurable dimension types are editable, and system-defined dimension types are not editable.

Configurable dimension types include entity, scenario, account, flow, user defined, parent, intercompany, time, and consolidation. See Configurable Dimensions.

System-defined dimension types include origin and view. See System-defined Dimensions.

The following dimension tokens are used to refer to dimension types in OneStream.

Dimension Token	Dimension Type
E#	Entity
S#	Scenario
A#	Account
F#	Flow
U1#	User defined 1

Dimension Token	Dimension Type
U2#	User defined 2
U3#	User defined 3
U4#	User defined 4
U5#	User defined 5
U6#	User defined 6
U7#	User defined 7
U8#	User defined 8
P#	Parent
I#	Intercompany
T#	Time
C#	Consolidation
O#	Origin
V#	View

Configurable Dimensions

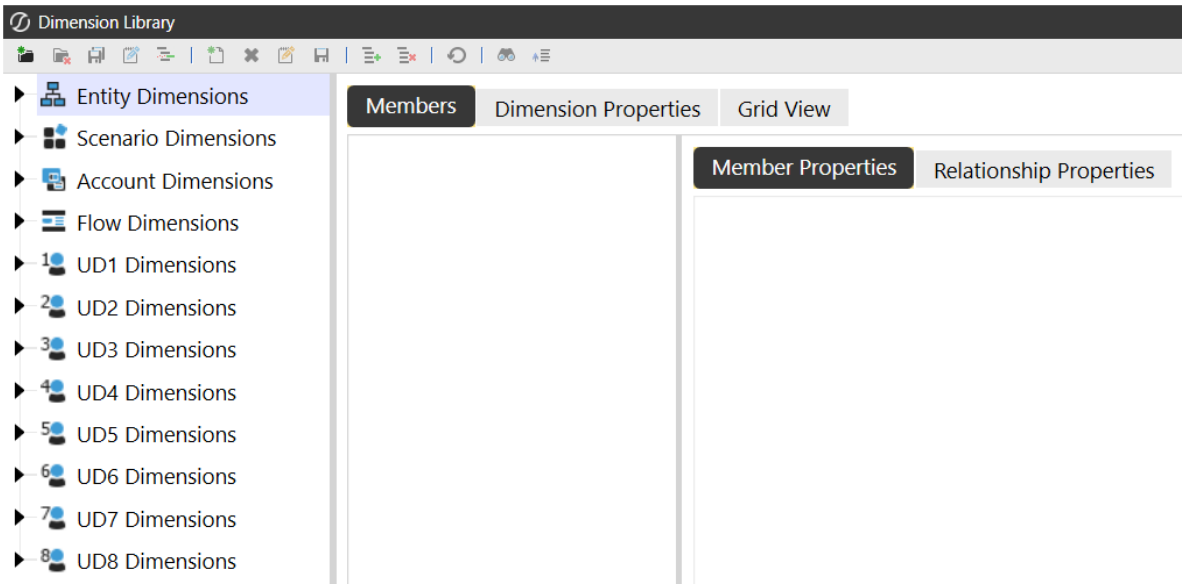
Configurable dimension types include entity, scenario, account, flow, user defined, parent, intercompany, time, and consolidation.

- Entity, scenario, account, flow, and user defined dimensions are created, maintained, and edited in the Dimension Library.
- Parent and intercompany dimensions are derived based on the entity dimension and its properties.
- Before an application is created, the time dimension can be customized in **System** > **Time Dimensions** and **Application** > **Time Profiles**. After an application is created, the time dimension cannot be edited, but it can be viewed using the point of view (POV) pane.
- The consolidation dimension has currencies that are added to Application Properties. The consolidation dimension is not editable in the Dimension Library, but it can be viewed using the POV pane.

Dimension Library Navigation

The Dimension Library enables all the customizable dimensions to be defined for your business needs. Dimensions can be shared across multiple cubes. Defining dimensions is important for sharing between cubes.

The entity, scenario, account, flow, and user defined dimensions are created and maintained in the **Dimension Library** in **Application** > **Cube** > **Dimensions**.



The following icons are in the Dimension Library toolbar:

- **Create Dimension**: Creates a new dimension
- **Delete Selected Dimension**: Deletes a dimension after its members are deleted
- **Save Dimension**: Saves changes to the dimension properties
- **Rename Dimension**: Renames a dimension

  > **NOTE:** If you rename a dimension, you need to update objects that reference that dimension. For example, cube, data sources, transformation rules.

- **Move Dimension**: Moves a dimension. You can move a dimension one level above a parent or below a sibling.

> **NOTE:** This action is not supported for entity dimensions, but can be used for account, scenario, flow, and user defined dimensions.

-  **Create Member**: Creates a new member for the selected dimension

> **NOTE:** The maximum member name length is 500 characters.

-  **Delete Selected Member**: Deletes a selected member that has no data stored

-  **Rename Selected Member**: Renames a selected member in the Dimension Library only

> **NOTE:** It does not rename the member in other areas.

-  **Save Member**: Saves changes to the member properties

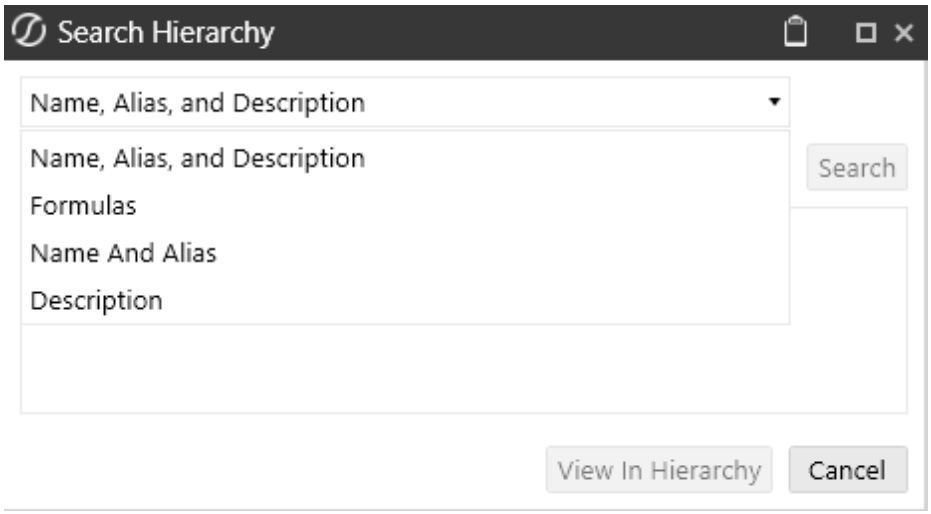
-  **Add Relationship for Selected Member**: Moves a member to another hierarchy or places it in more than one hierarchy

-  **Remove Selected Relationship Without Deleting Member**: Removes a member from a hierarchy without deleting the selected member. This can also be done by right-clicking on a dimension member.

-  **Cancel All Changes Since Last Save**: Cancels changes made to the member and dimension properties since the last save

- **Search Hierarchy**: Searches for a member within a dimension member hierarchy.

  This search will produce every hierarchy in which a member appears. Search using Name, Alias, and Description, Formulas, Name And Alias, or Description.



- **Collapse Hierarchy**: Collapses a member hierarchy within a dimension

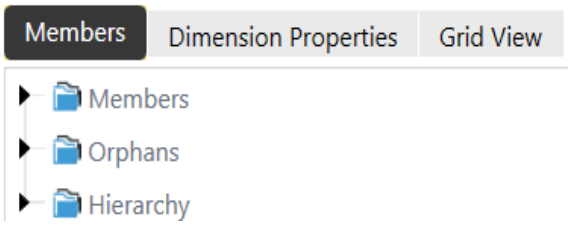
The following icon displays in all dimension security properties.

- **Navigate to Security**: Navigates to the Security page. Use this icon to make changes to security users or groups before assigning them to specific dimensions.

Members Tab

To view the **Members** tab, go to **Application** > **Cube** > **Dimensions** and select a dimension member. Then, select the **Members** tab.

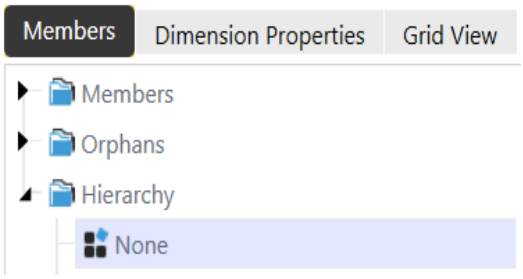
In the Members tab, dimensions can be viewed by members, orphans, or in a hierarchy.



- **Members**: All members within the selected dimension, including orphans.
- **Orphans**: Members not assigned to a dimension hierarchy.
- **Hierarchy**: Members in the chosen dimension in their hierarchy.

Every dimension hierarchy has a **None** member. The **None** member is used when no selection for the dimension is needed or applicable.

> **Example:** The **None** member is referenced in the POV pane if the dimension is not needed in the analysis or data load.



Member Properties and Relationship Properties Tabs

Dimensions contain members that can be inherited and extended through Extensible Dimensionality. Use the **Add Relationship for Selected Member** icon to move a member to another hierarchy or place it in more than one hierarchy.

Dimension members have member properties and relationship properties tabs. The properties on these tabs vary by dimension type. Depending on the field, you can edit the properties using the following methods:

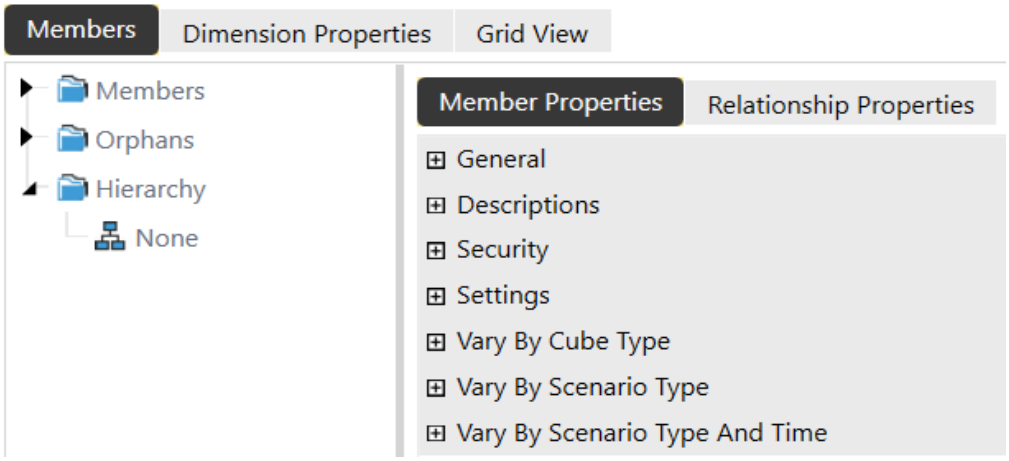
- Typing directly in the field
- Selecting the ellipsis to open a dialog box
- Selecting the **Navigate to Security** icon in the Security section
- Double-clicking in a field to generate a drop-down menu

Member Properties Tab

Member properties are settings specific to the individual member. These properties apply to the member regardless of its relationship to a particular parent member in the hierarchy. If a member is part of multiple hierarchies, updates to the member properties apply to the member as it displays in all hierarchies regardless of which relationship is selected when the member properties are updated.

Member properties are grouped into categories. For example, General, Descriptions, and Security).. Various property fields are included in each category. Both the categories and their property fields vary across the dimension types. The plus and minus icons next to each category can be used to expand or collapse it.

To view the **Member Properties** tab, go to **Application** > **Cube** > **Dimensions** and select a dimension member. Select the **Members** tab and then the **Member Properties** tab. For example, this image shows the entity dimension.



These properties are standard across the entity, scenario, account, flow, and user defined dimensions:

- General
- Descriptions

General

- **Dimension Type**: Indicates which dimension is being used.
- **Dimension**: Indicates the dimension name.
- **Member Dimension**: The dimension to which it is a member.
- **Id**: A numerical identity that is automatically assigned by the application when the member is created. This number cannot be changed, even if the member name is changed.
- **Name**: The name of the member in the dimension.

- **Alias**: Use this property to add aliases to dimension members. Aliases are optional names that you can use as alternative ways to reference members from areas such as Member Filters, Cube Views, or Business Rules. This enables you to use additional, readable names for dimension members in new or existing solutions, without disrupting existing solutions that reference the main dimension member names. For example, you can add the original dimension member name as an alias before renaming the dimension member name to ensure existing solutions referencing the dimension member do not break after the name change. Multiple aliases can also be added to the same dimension member through a comma-delimited list.

  > **NOTE:** You cannot have duplicate alias names in the same dimension type. You cannot create an alias with the same name as a different dimension member within the same dimension type, and you cannot create a new member with the same name as an existing alias within the same dimension type.

The following properties only display for a Dynamic Dimension:

- **External Member Id**: A numerical identity for the external member.
- **External Member Name**: The name of the external member in the dynamic dimension.

Descriptions

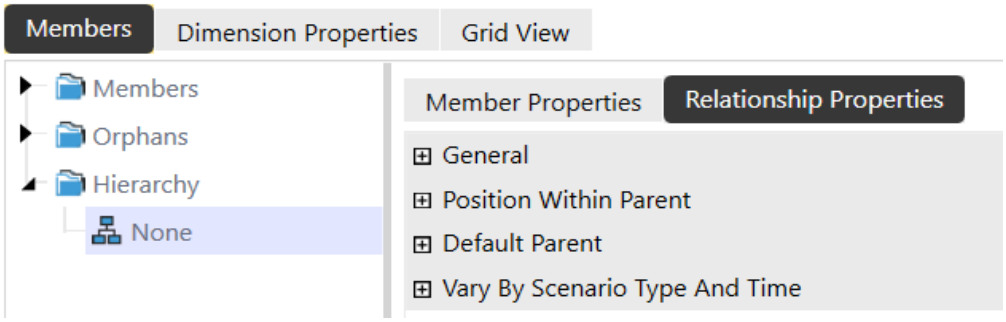
Default Description: A description of the member in the dimension. . See "Report Alias" in the *System Administration Guide*.

If the Application Server Configuration File has been updated to include additional cultures, other language description fields may display under the Default Description field.

Relationship Properties Tab

Relationship properties are settings associated with the relationship between members in the hierarchy. Relationship properties can vary by each parent-child relationship, so different behaviors can occur based on the selected relationship.

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** and select a dimension member. Select the **Members** tab and then the **Relationship Properties** tab. For example, this image shows the entity dimension.



These properties are standard across the entity, scenario, account, flow, and user defined dimensions:

- General
- Position Within Parent

General

- **Dimension Type**: Indicates which dimension is being used.
- **Dimension**: Indicates the dimension name.
- **Parent Member Dimension**: Displays the name of the parent member. This may be blank if it is at the root level.
- **Member Dimension**: Displays the name of the member.

- **Parent Member Name**: Displays the direct parent to the member. This may display Root if it is the first branch.
- **Member Name**: Displays the name of the member as it was defined.

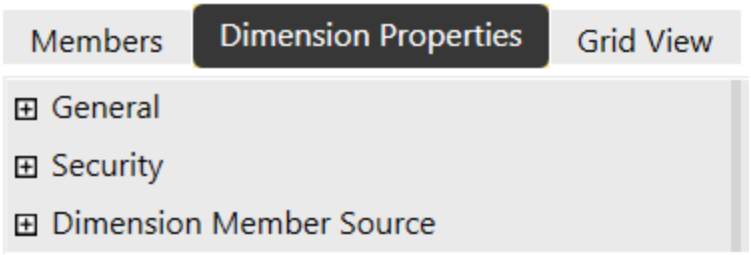
Position Within Parent

When a new member is created, it is added to the bottom of the list of its siblings.

- **Position**: Use this field to change the current level of the member in the hierarchy in relation to its siblings. This setting cannot move a member out of the current hierarchy.
- **Sibling Member**: Displays a list of all the siblings in the current hierarchy available to move based on the setting in the Position field.

Dimension Properties Tab

To view the **Dimension Properties** tab, go to **Application** > **Cube** > **Dimensions** and select a dimension member. Then select the **Dimension Properties** tab.



These properties are standard across the entity, scenario, account, flow, and user defined dimensions:

- **Dimension Type**: Indicates the current dimension type.
- **Id**: A numerical identity that is automatically assigned by the application when the dimension is created. This number cannot be changed, even if the dimension name is changed.
- **Name**: Indicates the current name of the dimension.
- **Description**: Indicates the current description of the dimension.
- **Access Group**: Displays the security group that has access to the dimension.
- **Maintenance Group**: Displays the security group that has access and can make changes to the dimension.
- **Source Type**: The approach used to specify members for this dimension. For example, members and relationships for a Standard dimension are created manually using the Dimension Library Editor or by using XML load. Members and relationships for a Dynamic dimension can be sourced from external data using a custom implementation in a Workspace Assembly
    - **Standard**: Normal metadata member must match a cube.
    - **Business Rule**: Connects to a business rule so that a set of members does not have to match a cube.
    - **Dynamic Dimensions**: Sources members and relationships for a Dynamic Dimensions to cache metadata from an external source into OneStream using Workspace Assemblies.
    - **XBRL**: Connects to an XBRL (extensible business reporting language) taxonomy. XBRL is a language for the electronic communication of business and financial data worldwide.

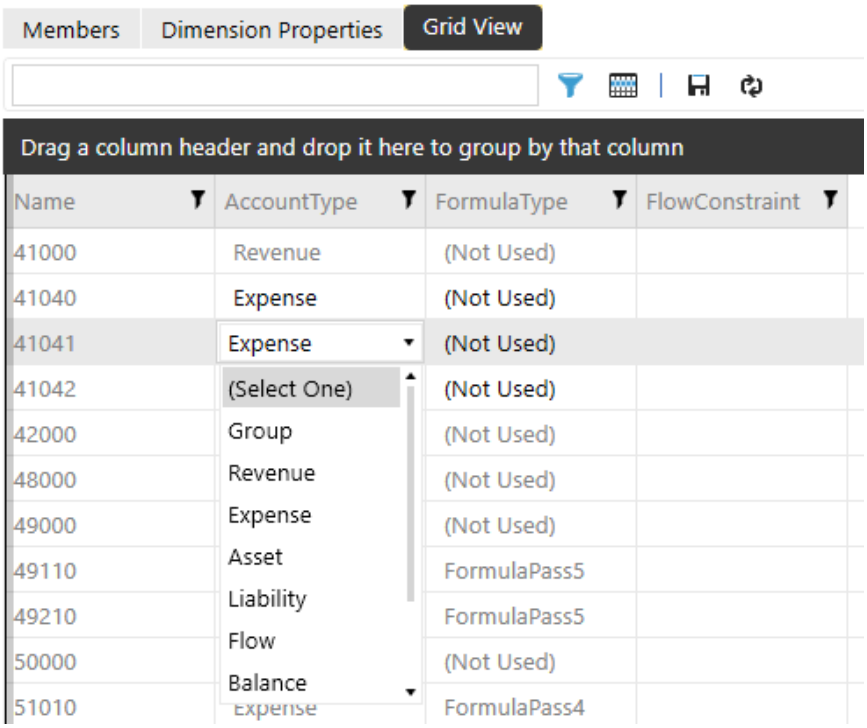
- **Source Path**: If the Source Type is Business Rule or XBRL, enter the business rule name or the full file system path for the XBRL taxonomy link definition file. For example, Internal/XBRL/Taxonomies/EntryPoints/. You can also enter a Workspace Assembly Service to run for a Dynamic Dimension. For example, Workspace.DynamicDimensionsWS.WS.
- **Name Value Pairs**: If the Source Type is Business Rule or XBRL, enter a comma-separated list of name-value pairs.

Grid View Tab

Select the **Grid View** tab to view a complete list of members from a selected dimension. You can view and edit properties of multiple members in the same screen without navigating to one member at a time. The following icons are in the Grid View toolbar:

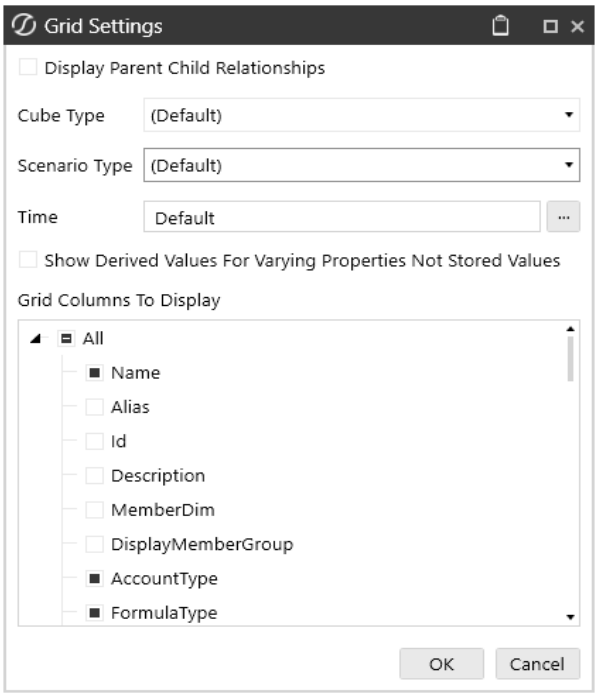
- **Member Filter**: Narrows the list to view specific members. See "Member Filter Builder" in the *Presentation Guide*.
- **Grid Settings**: Changes the data displayed.
- **Save**: Saves changes.
- **Refresh**: Updates the data.

To change the metadata settings of multiple members in this view, double-click a data cell.



Grid Settings

Members displayed can change based on cube type, scenario type, and the time member chosen. **Grid Columns to Display** lists all the member properties available. Select the ones to display and edit in the grid.



1. Select **Grid Settings** to change the data displayed.
2. In **Grid Settings**:
    a. Select **Cube Type** to view members in a specific cube.
    b. Select **Scenario Type** to view members from a certain scenario type.
    c. Select **Time** to view members using a specific time member. Settings can be turned on or off, and the formulas will change to look at a specific time frame.
    d. Select **Show Derived Values for Varying Properties Not Stored Values** to display metadata that varies by scenario and time.
    e. In **Grid Columns to Display**, select the member properties to view and edit in the grid.

Build a Dimension

To create a new dimension:

1. In the **Dimension Library**, select a dimension type.
2. Select the **Create Dimension** icon.
3. Enter a name.
4. Select the **OK** button.

Create a Member

To create a new member in the dimension:

1. Select the dimension.
2. Select the **Create Member** icon.
3. In the **Member Properties** tab, enter a name and default description. Add and update any additional properties as needed.
4. In the **Relationship Properties** tab, add and update as needed.
5. Select the **Save Member** icon.

Clone a Member

To create a new member using the same settings as the selected member.

1. In the **Members** tab, select a member and then right-click.
2. Hover over **Clone Member** to display the menu of options.
3. Select the option to clone the new member as: As First Child, As Last Child, As Previous Sibling, or As Next Sibling.

   > **NOTE:** Formula types and formulas do not copy to the new member.

4. In the **Member Properties** tab, enter a name and default description. Add and update any additional properties as needed.
5. In the **Relationship Properties** tab, add and update as needed.
6. Select the **Save Member** icon.

Change the Position of a Member

This option alters the position of a member in the hierarchy in relation to a sibling.

1. In the **Members** tab, select a member.
2. Select the **Relationships Properties** tab.
3. Select the **Position** drop-down menu.
4. Select the new position for the member: Retain Current Position, Before Sibling Member, After Sibling Member, First Sibling, or Last Sibling.
5. Select the **Sibling Member** drop-down menu.
6. Select the sibling member.
7. Select the **Save Member** icon.

Expand All Descendants

This option expands all the descendants for the selected member.

1. In the **Members** tab, select a member that has children and then right-click.
2. Select **Expand All Descendants**.

Copy and Paste Relationships

This option adds or moves copied members to a new hierarchy.

1. In the **Members** tab, select a member and then right-click.

   > **TIP:** Press **Ctrl** + click to select multiple items.

2. Select **Copy Selected Members**.
3. Right-click the name of another member.
4. Select the paste option:
   - Select **Paste Relationships (Add)** to create a new relationship for the copied members.
   - Select **Paste Relationships (Move)** to remove members from their current relationship and move them to a new one.
5. Select the location to paste the copied member: As First Child, As Last Child, As Previous Sibling, or As Next Sibling.

Add Relationship for Selected Member

This creates a relationship between two members.

1. In the **Members** tab, select a member.
2. Select the **Add Relationship For Selected Member** icon.

3. Select the ellipsis next to **Child Member** to select the child member.
4. Select the **OK** button.

Remove Relationships

This option removes members from their current relationship without moving them to a new one. If the member is no longer part of the dimension structure, it will be placed under **Orphans**. This option will not delete the member.

1. In the **Members** tab, select a member.
2. Select the **Remove Relationships** icon.

> **NOTE:** You can also select a member and then right-click to see an option for **Remove Relationships**.

Delete a Member

This option deletes the selected member from the Dimension Library. A member can only be deleted if it does not have data stored.

1. In the **Members** tab, select a member.
2. Select the **Delete Selected Member** icon.

> **TIP:** You can also select a member and then right-click to see an option for **Delete Member**.

Dimension Member Names

Use the following guidelines when creating dimension member names:

- Names must be unique within a dimension type.

  > **Example:** There can be only one account named GrossIncome across all account dimensions.

- Names have a 500 character limit.
- Use underscores in place of spaces and periods. Square brackets do not need to be used if a member name has an underscore.
- Spaces are allowed but not recommended.
- If a member name includes a space or a period, square brackets must be used if the member is queried.

  > **Example:** E#[Quebec.City]

Restricted Characters

Restricted characters should not be used in dimension member names:

Character	Name and Description
&	Ampersand **NOTE:** The ampersand character can be used in member names, but it may affect platform functionality. So, it is recommended to not use it.
*	Asterisk
@	At sign

Character	Name and Description
\	Back slash
{ }	Brace (left and right)
[ ]	Bracket (left and right)
^	Caret
,	Comma
=	Equal sign
!	Exclamation point
>	Greater than sign
<	Less than sign
-	Minus sign
#	Number sign
%	Percent **NOTE:** The percent character can be used in member names, but it may affect platform functionality. It is recommended to not use it. The percent character cannot be used in an alias name.

Character	Name and Description
|	Pipe or vertical bar
+	Plus sign
?	Question mark
"	Quotation marks
;	Semicolon
/	Slash

Reserved Words

Reserved words cannot be used in dimensions or other structural application components, such as cubes.

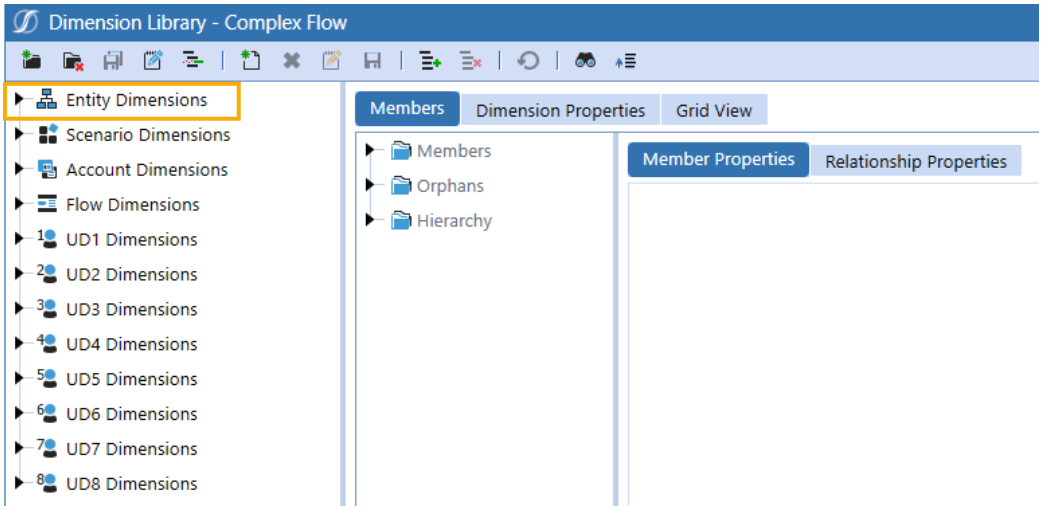
- Account
- All
- Cons
- Consolidation
- Default
- DimType
- Entity
- EntityDefault

- Flow Origin
- IC
- None
- Parent
- POV
- Root
- RootAccountDim
- RootEntityDim
- RootFlowDim
- RootScenarioDim
- RootUD1Dim
- RootUD2Dim
- RootUD3Dim
- RootUD4Dim
- RootUD5Dim
- RootUD6Dim
- RootUD7Dim
- RootUD8Dim
- Scenario
- Time
- UD1 – UD8

- UD1Default
- Unknown
- View
- WF
- Workflow
- XFCommon

Entity Dimension

The entity dimension represents the business unit structure of an organization. In a multi-cube application in which each cube contains different entities, the cubes can be used to link the different entity structures.



Member Properties Tab

To view the **Member Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Entity Dimensions** and select an entity member. Select the **Members** tab and then the **Member Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Descriptions

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Security
- Settings
- Vary By Cube Type
- Vary By Scenario Type
- Vary By Scenario Type and Time

Security

Display Member Group: This group can see that this entity exists within a list of entities.

> **NOTE:** Display member security limits the data that displays in intercompany matching reports. To display intercompany transactions, users must be in the display member groups of the entity and the counterparty.

Read Data Group: This group can see data from this entity.

Read Data Group 2: This is a second group that can see data from this entity. It is used for additional security details.

Read and Write Data Group: This group can view and edit data from this entity.

Read and Write Data Group 2: This is a second group that can view and edit data from this entity. It is used for additional security details.

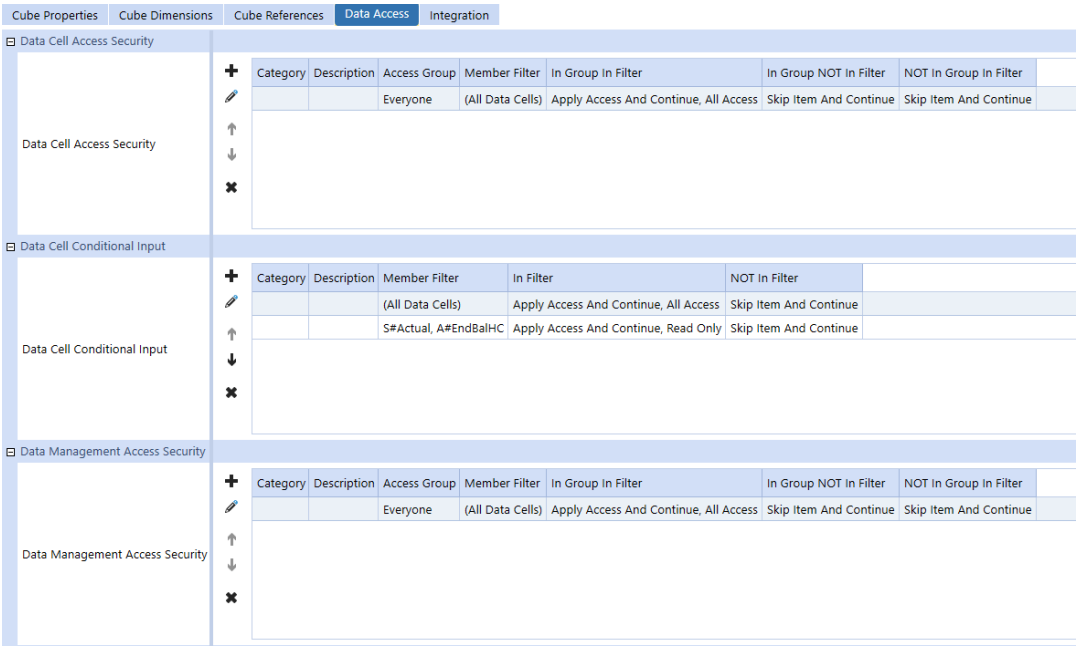
> **TIP:** Select the ellipsis button and begin typing the name of the security group in the blank field. As the first few letters are typed, the groups are filtered making it easier to find and select the desired group. Once the group is selected, press **Ctrl** + double-click to enter the correct name in the field.

Use Cube Data Access Security

- **True**: Cube Data Access Security will be applied from the cube level down to this entity.
- **False**: Cube Data Access Security will not be applied.

Cube Data Cell Access Categories, **Cube Conditional Input Categories**, and **Cube Data Management Access Categories**: Category is used to specify an optional name for a group of Cube Data Access Items. Multiple items can use the same category name. These settings are used when the data cell security for an entity should only use a subset of the cube data access items. Enter a comma-separated list of category names.

If these settings are left blank, all categories will be used in the corresponding settings in the Data Access tab in **Application** > **Cube** > **Cubes** > **Data Access**. See Data Access Tab.



Settings

Currency: The local currency of the entity.

Is Consolidated

- **True**: The data from the children of this entity are consolidated, which results in the entity equaling the total of its children.
- **False**: The data will not be consolidated. The use case to set the Is Consolidated property to false is to use the parent entity strictly for grouping purposes. Also, it can help with consolidation performance times, because the consolidation will not be performed at the parent entity.

Is IC Entity

- **True**: Makes the entity visible in the intercompany dimension and allows that entity to post intercompany transactions to intercompany accounts.
- **False**: This will not be an intercompany entity.

Vary By Cube Type

Each cube type can use different settings, because dimensions can belong to multiple cubes.

Constraints are properties used on given members to stop them from interacting with other members of another dimension type.

Flow Constraint: This is the flow dimension constraint. This entity can only use the members with this child or members under a selected parent member.

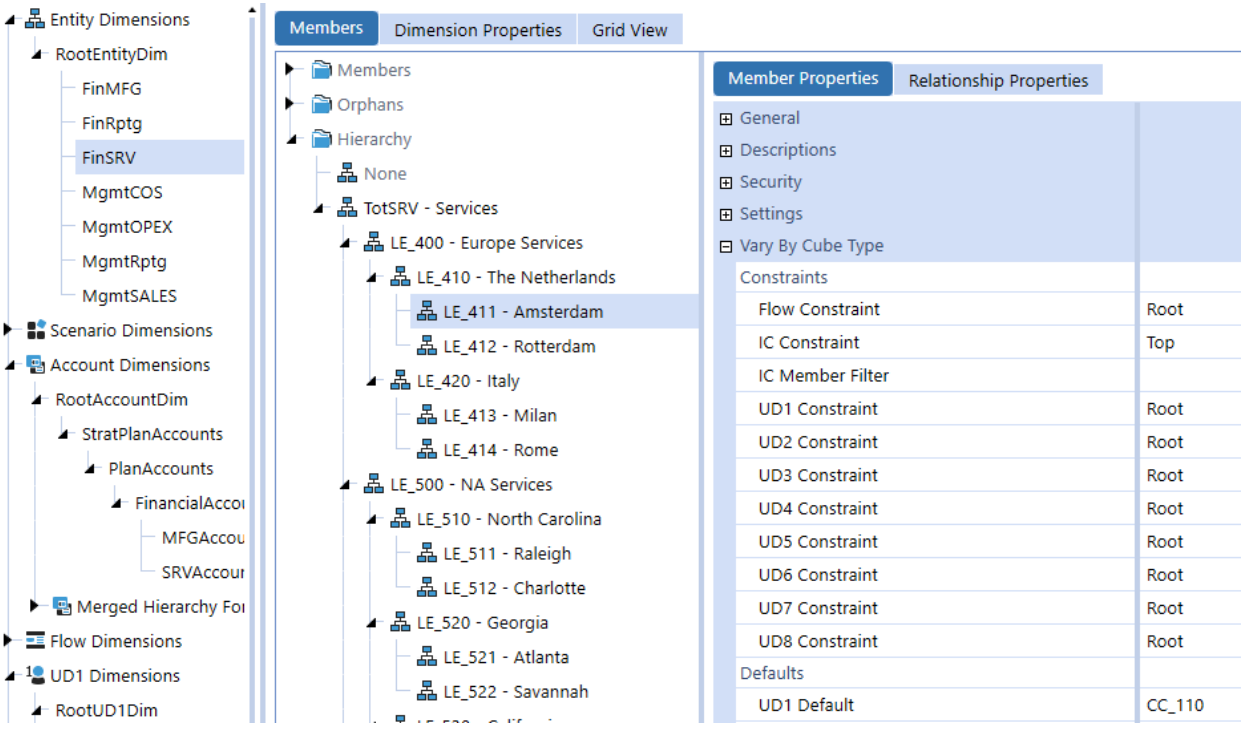
IC Constraint: This is the intercompany dimension constraint. This entity can only use the members with this child or members under a selected parent member. Setting entity constraints will define the data intersection as green, no input cells.

IC Member Filter: This is an additional way to limit intercompany partners of a particular entity. Use the IC Member Filter to make a list of partners. The entity can only have intercompany transactions with this list of partners. Conversely, only this list of partners can have a transaction with the entity. This provides additional protection to the intercompany transaction.

UD1 Constraint through **UD8 Constraint**: This is the user defined constraint. This entity can only use the members with this child or members under a selected parent member.

UD1 Default through **UD8 Default**: A default membership for this entity can be defined to a user defined dimension, such as a cost center.

Example: This entity falls in the Europe Services region and every import, form edit, and journal entry should classify this entity as being associated with the CC_110 - Sales cost center. The entity never has to explicitly map to the cost center. When data is loaded, it is directed to the EntityDefault member and the user defined setting will be applied automatically. This can have a negative impact on the consolidation if not used in a user defined dimension with a limited number of members.



Vary By Scenario Type

Sibling Consolidation Pass: This is typically used for holding companies related to Equity Pickup calculations. Specify Pass 2 or greater if calculated data for this entity is based on calculated data from other sibling entities during a consolidation. It allows this entity to be consolidated and calculated after other sibling entities have been consolidated and calculated. The default behavior for all entities is either Pass 1 or (Use Default). Both settings do the same thing, effectively causing all sibling entities to be consolidated (if they have child entities) and calculated at the same time or in an indeterminate order.

Sibling Repeat Calculation Pass: This is typically used for circular ownership related to Equity Pickup calculations. Specify Pass 1 or greater to repeat the calculation of the local consolidation member of this entity. Repeat calculation passes occur after all sibling entities have been consolidated and calculated. Repeat calculations are used when calculated data for two entities rely on the calculated data of each other. The default behavior for all entities is (Use Default), which will not use a repeat calculation.

Auto Translation Currencies: This is typically used for an Equity Pickup calculation when an entity needs to be translated to a sibling holding the local currency of a company during a consolidation. Enter a comma-separated list of currencies. The default behavior is to translate only to the local currency of the parent entity during a consolidation.

See Equity Pickup.

Vary By Scenario Type and Time

Entities can have different attributes based on scenario and time. Default settings are applied from the first time period in the application until there is a change, and some settings vary by scenario type. The time display for any time varying properties uses the standard time dimension profile.

In Use

- **True**: The entity is in use.
- **False**: This can turn off the ability to use an entity based on time. This keeps historical data available. This is designed to be used when an entity becomes inactive or is sold. Once an entity is no longer in use, it will be ignored during consolidation and all intersections including this entity will be not valid.

Allow Adjustments: This setting can be applied as True or False, or it can vary by scenario type or time.

- **True** (default): The Journals Module is enabled for the entity to enter adjustments to the AdjInput Origin member.
- **False**: The Journals Module is not enabled for the entity. However, when False, adjustment to the AdjInput Origin member is still allowed on accounts having the Account Adjustment Type of Data Entry, which is used in designs where adjustments are performed using form data entry rather than the Journals Module. To prevent input to AdjInput on accounts set as Data Entry Adjustment Type, NoInput Rules, or Data Cell Conditional, security can be used.

Allow Adjustments From Children: This setting can be applied as True or False as a default, or it can vary by scenario type or time. This setting is applied on parent-level entities to allow the direct child entities to post journal adjustments using the consolidation dimension members OwnerPostAdj and OwnerPreAdj. If set to **True** (default), adjustments from the direct child entities to OwnerPostAdj and OwnerPreAdj are allowed.

Text 1 through **Text 8**: This is open for custom attributes used for multiple purposes, such as business rules, member filters, or transformation rules. The value can be changed as the business changes or by scenario type.

Relationship Properties Tab

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Entity Dimensions** and select an entity member. Select the **Members** tab and then the **Relationship Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Position Within Parent

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Default Parent
- Vary By Scenario Type and Time

Default Parent

Parent Sort Order: Setting to determine a default parent when evaluating member lists, for example, in cube views. If a parent is not explicitly specified, the parent with the lowest sort order is used.

Vary By Scenario Type and Time

Percent Consolidation: Defines the percentage of the entity to be consolidated.

> **Example:** A value of 100 means that 100% of the data will be consolidated to the parent. A value of 50 means that 50% of the value will be consolidated to the parent.

Percent Ownership: Ownership setting that can be used by business rules. By itself, the setting has no effect on the consolidation.

Ownership Type: Ownership setting that can be used by business rules. By itself, the setting has no effect on the consolidation.

- **Full Consolidation**: Normal setting for entities that fully consolidate into a parent.
- **Holding**: Designates the parent and child relationship as a holding company situation.
- **Equity**: Used to help business rules determine the value to increase the equity method of accounting for an investment.
- **Non-Controlling Interest**: Used for business rules to determine the minority interest portion of an entity into the consolidation.
- **Custom**: Open for custom use in business rules.

Text 1 through **Text 8**: Use to define custom attributes to modify aspects of business rules, member filters, or transformation rules capabilities. These custom attributes act as variable placeholders activated at runtime to customize views of data or ways to interact with data. You can change these values by scenario type or for evolving business needs.

Scenario Dimension

The scenario dimension includes members representing a set of data that differs by business use, such as actual, budget, and forecast.

Scenario Types

Members are assigned a scenario type and each cube offers the following preset scenario types:

- Actual
- Administration
- Budget
- Control
- Flash
- Forecast
- FXModel
- History
- LongTerm
- Model
- Operational
- Plan
- Sustainability
- Target
- Tax
- Variance
- ScenarioType1-8
- Default

Scenario types are predefined scenario categories used in conjunction with varying member properties, data sources, transformation rules and workflow profiles. Different versions for each of the preset types can be created depending on which scenario type is assigned to the current scenario being processed. Scenario types provide you flexibility since each type contains an unlimited number of scenarios. You can assign dimensions to cubes, differing them by scenario type.

Scenarios can be organized as a hierarchy, but do not consolidate.

Names of scenario types are irrelevant since none of the default behaviors of the OneStream Platform are defined by scenario types. This enables you to choose any scenario type, assign it to any scenario, and choose a relevant name applicable to the business problem you're solving.

> **NOTE:** The name and number of the scenario types cannot be changed. Multiple scenarios can share the same scenario type. When shared, all the features will work the same way across all scenarios sharing the same scenario type.

Member Properties Tab

To view the **Member Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Scenario Dimensions** and select a scenario member. Select the **Members** tab and then the **Member Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Descriptions

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Security
- Workflow
- Settings
- FX Rates
- Hybrid Source Data
- Custom Settings

Security

Use security settings to control the level of access organization members have to a scenario.

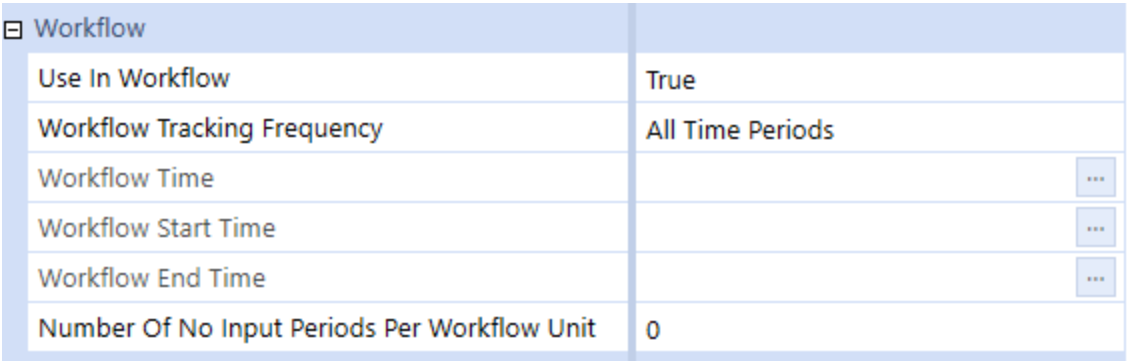


Select the ellipsis to select the level of access for each group.

- **Read Data Group**: Controls who can view data
- **Read and Write Data Group**: Determines who can view and modify data
- **Calculate from Grids Group**: Sets who can calculate, translate, and consolidate data from a Cube View or form
- **Manage Data Group**: Determines who can run data management tasks. Membership in this group is required to run steps, such as custom calculate or reset scenario. This prevents unauthorized users from launching steps that could alter or clear data unintentionally.

Workflow

Use workflow settings to control the type of periods displayed to users when they load data.



Use in Workflow: Determines whether a scenario is viewable from the workflow in OnePlace.

- Set to **False** to make the scenario unavailable.
- Set to **True** to make the scenario available to users setting the workflow point of view.

> **NOTE:** Data can still be entered with forms and the Excel Add-In to a hidden scenario.

Workflow Tracking Frequency: Determines how time displays in the workflow and is based on the type of data being entered for the scenario.

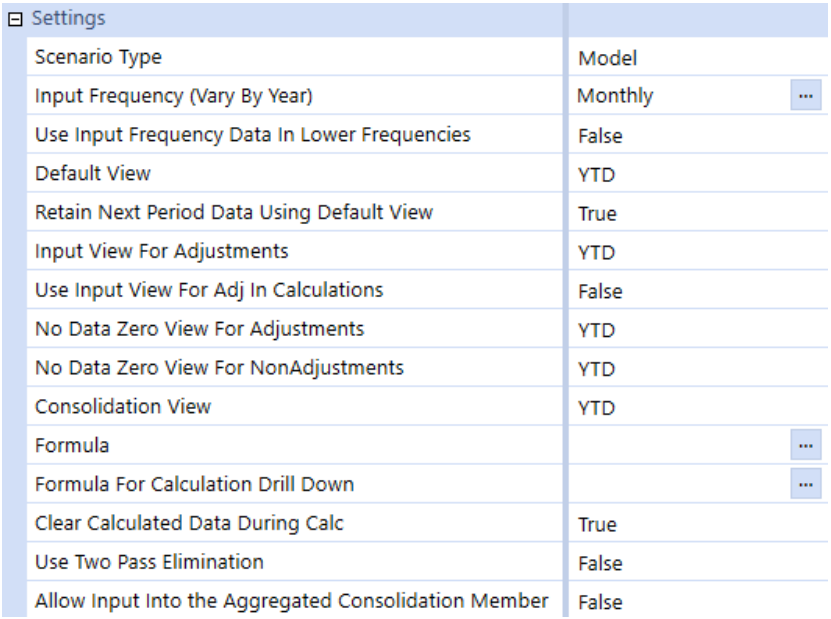
> **NOTE:** Workflow Tracking Frequency settings cannot be changed if any steps in the workflow have been processed for the scenario.

- Choose from the following options:
  - **All Time Periods** (standard setting): To display all periods in the year. A period could be months or weeks depending on the application
  - **Monthly**: To set the workflow periods to monthly, which could be 12 to 16 months depending on the application.
  - **Quarterly**: To set the workflow periods to four periods, such as Q1, Q2, Q3, Q4
  - **Half Yearly**: To set the workflow periods to two periods, such as H1, H2
  - **Yearly**: To set the workflow periods to one period, such as 2023
  - **Range**: To define a custom range that is displayed as a one-time period including the start and end time, such as 2023M12-2024M11. As data loads, each period displays this range.
    - **Workflow Time**, **Workflow Start Time**, and **Workflow End Time** become selectable when **Range** is chosen. Sellect the **ellipsis** to choose a time for each. Year or year and month can be defined.

Number of No Input Periods Per Workflow Unit: Determines the number of periods to be no input and disables input from the Import, Form, and Adjustment Origin Members for the specified periods. To set the number of no input periods, type a number in the cell.

> **Example:** If the first three months of data is automatically copied from Actual to Forecast with a user inputting the remaining nine months, enter 3 to make those first three periods read only and prevent data entry.

Settings

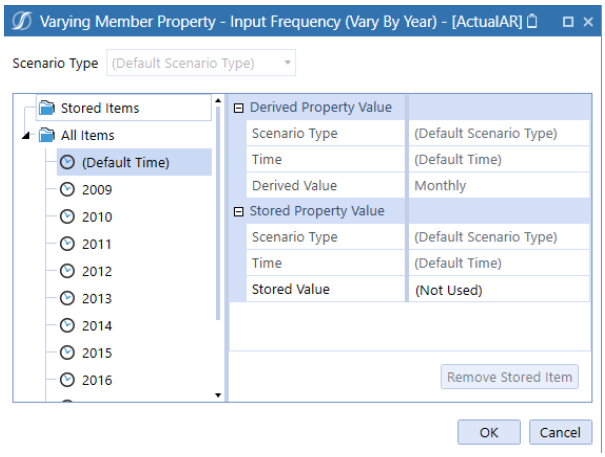


Settings	
Scenario Type	Model
Input Frequency (Vary By Year)	Monthly
Use Input Frequency Data In Lower Frequencies	False
Default View	YTD
Retain Next Period Data Using Default View	True
Input View For Adjustments	YTD
Use Input View For Adj In Calculations	False
No Data Zero View For Adjustments	YTD
No Data Zero View For NonAdjustments	YTD
Consolidation View	YTD
Formula	
Formula For Calculation Drill Down	
Clear Calculated Data During Calc	True
Use Two Pass Elimination	False
Allow Input Into the Aggregated Consolidation Member	False

Scenario Type: Groups similar scenario types to share settings or business rules. A scenario type can contain many scenarios.

Input Frequency (Vary By Year): Sets the data frequency to monthly, quarterly, half yearly, or yearly, for the scenario.

1. Select the **ellipsis** to vary the input frequency by year.
2. In the **Varying Member Property window**, from the pane on the left, choose the year.



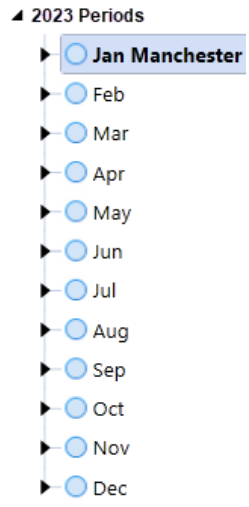
3. In the **Stored Value** field, select the frequency.
4. Select the **OK** button.

> **NOTE:** If Quarterly or Yearly is used, data is not saved at a base (monthly/weekly) level as this would display invalid cells.

The input frequency property works with the Workflow tracking frequency property. If the input frequency varies by year, the Workflow tracking frequency updates the workflow view accordingly. See examples below:

Example

If in 2023 the input frequency is monthly and the Workflow tracking frequency is All Time Periods, the workflow displays the following:



Example 2

If in 2024 the input frequency is weekly, and the Workflow tracking frequency is All Time Periods, the workflow displays the following:



Use Input Frequency Data in Lower Frequencies: Used to display data from the scenario's input frequency when processing a lower level invalid frequency.

> **Example:** In a quarterly scenario, display the amount from Q1 in M1, M2, M3 and in all weeks below these 3 months. In a yearly scenario, display the single yearly amount in all time periods in the year.

- Set to **True** to enable.
- Set to **False** to disable.

Default View: Used to set the standard view, either periodic or YTD, for calculations, member formulas, and clearing calculated data.

Retain Next Period Data Using Default View: Determines whether a future period's data value remains the same or is updated based on the prior period's input value. The scenario's default view is either periodic or YTD.

- Set to **True** if a flow account has a data value in a future period that will change if the data is altered in a prior period.
- Set to **False** if a flow account has a data value in a future period that will be retained if a data value in a prior period is being changed.

> **Example:** If set to true, February periodic and YTD data will need to change to be consistent with a new January entry. If false, when a new periodic or YTD data value is entered for January, February's data value is retained.

Input View For Adjustments: Determines the standard view on how data is entered using a journal. This typically matches the default view, either periodic or YTD.

> **NOTE:** Regardless of the setting, all journals, except auto-reversing, must be entered monthly.

Use Input View for Adj in Calculations: This property forces calculations to use the Input View For Adjustments view even if the calculation attempts to override the default view.

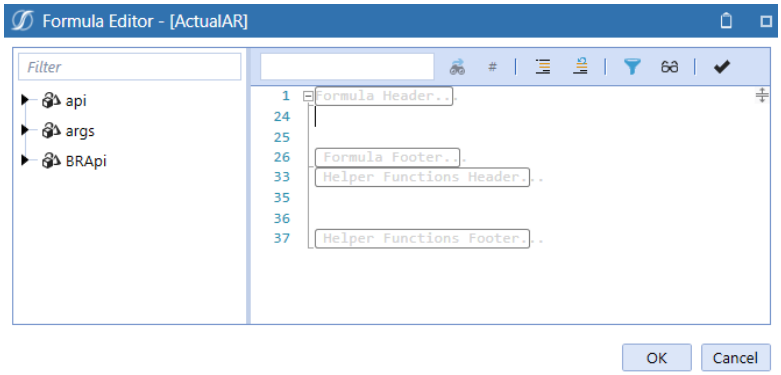
- Set to **True** to use the Input View for Adjustments view regardless of the default scenario view or the view specified in the calculation.
- Set to **False** to use the default scenario view when the view is not specified in the calculation.

No Data Zero View For Adjustments: Determines the view when there is no data in journals for the period. Can be set to periodic or YTD and typically the setting is the same as the Input View for Adjustments.

No Data Zero View For NonAdjustments: This is tied to the data load and is used when there is not data for the period. The current selections are YTD and periodic.

Consolidation View: This property determines the standard view of the consolidation and can be set to periodic or YTD.

- Set to **Periodic** if work is completed by period. However, all numbers are stored as YTD. Set to periodic for Org by Period implementations.
- Select **YTD** to enhance consolidation performance. This property can be changed and will update for the next consolidation.
- **Formula**: Provides the ability to use a formula to run copying between scenarios prior to the consolidation of a Scenario. See Formula Guide in About the Financial Model for more details.



Formula for Calculation Drill Down: Enables drill down on members with attached formulas. A specific formula is required for drill down to display the original formula and the members that activate the drill down to the original values. Drill down on calculated data cells and data cells copied by a copy data management sequence. See "Formulas for Calculation Drill Down" in the *Application Tools Guide* for more information.

Clear Calculated Data During Calc: Determines whether existing data is cleared during the calculation process.

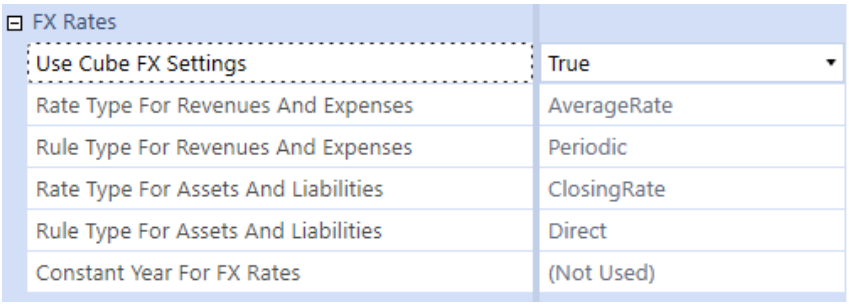
- Set to **True** (default), to clear data during the calculation process.
- Set to **False** to clear data manually.

Use Two Pass Elimination: Determines whether the use of the IsSecondPassEliminationCalc argument is recognized in Business Rules. The IsSecondPassEliminationCalc argument functions within a parent node to ensure the immediate sibling entities,within the node are fully calculated at Eliminations. The execution of the IsSecondPassEliminationCalc argument allows a sibling to retrieve elimination values from other siblings within the node.

Allow Input Into the Aggregated Consolidation Member: Determines whether data can be input into the consolidation member aggregation.

FX Rates

Use FX rates to determine the exchange rate settings for the selected scenario member.



FX Rates	
Use Cube FX Settings	True
Rate Type For Revenues And Expenses	AverageRate
Rule Type For Revenues And Expenses	Periodic
Rate Type For Assets And Liabilities	ClosingRate
Rule Type For Assets And Liabilities	Direct
Constant Year For FX Rates	(Not Used)

Use Cube FX Settings: Determines whether the scenario member uses the cube's default rate type.

- Set to **True** to use current cube's default rate type.
- Set to **False** to use custom rate calculations.

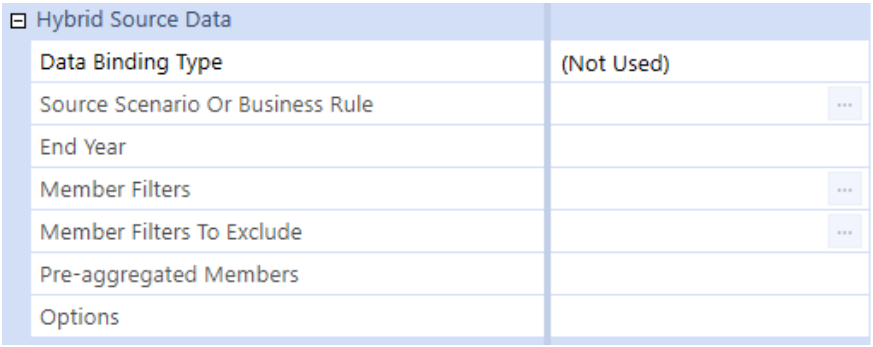
If **False** is selected, Rate and Rule Type for Revenues And Expenses, Rate and Rule Type for Assets And Liabilities, and Constant Year For FX Rates become accessible.

- **Rate Type for Revenue And Expenses** / **Rate Type For Assets And Liabilities**:
  - Choose **AverageRate** to use the average currency rate of a period from the first day to the last day of the month.
  - Choose **OpeningRate** to use the currency rate at the beginning of the period.
  - Choose **ClosingRate** to use the currency rate at the end of a period.
  - Choose **HistoricalRate** to use the opening currency rate from a particular historical date.
- **Rule Type For Revenue And Expenses** / **Rule Type For Assets And Liabilities** selections:
  - Set to **Direct** to multiply each value by the translation rate.
  - Set to **Periodic** to weight the calculation based on a period.

- **Constant Year for FX Rates**: Use the drop-down menu to set the rates based a specific year.

Hybrid Scenarios

Hybrid scenarios allow for data to be copied or shared between scenarios for defined period and specific members.



For information about how the following properties impact hybrid scenario queries and using hybrid scenarios, see Working With Hybrid Scenarios

Data Binding Type: Indicates the type of data binding that should occur for the target scenario. Choose one of the following options:

- **Share Data from Source Scenario** to share a read-only cube data set from the source scenario to the target scenario.
- **Copy Input Data from Source Scenario** to copy base level cube data from a source scenario to the target scenario.
- **Copy Input Data from Business Rule** to copy base level cube data based on a finance business rule.

Source Scenario or Business Rule: Indicates source location. Set to either the **source scenario member** or **business rule**.

End Year: Property controlling time.

Select a year if the data query must end after a specific year or leave the field empty to query all years.

> **Example:** If it should not occur in 2021, the End Year is set to 2020 to exclude all future years from the query.

> **NOTE:** The database stores data records by year, each year having its own data table and containing the data records for each period. When data is queried using a Hybrid Scenario, it occurs at the database level, and only returns the periods containing data.

Member Filters: Determines which data is included in query results.

- Use a comma separated list to include only members in the lists. This can include multiple dimension types, member expansions, or single members.
- Leave the field blank to include all source data in query results.

Member Filters to Exclude: Determines which data is excluded in query results.

- Use a comma separated list to exclude members in the lists. This can include multiple dimension types, member expansions or single members. If it includes members from the Member Filters property, those members are excluded.
- Leave the field blank to exclude no data.

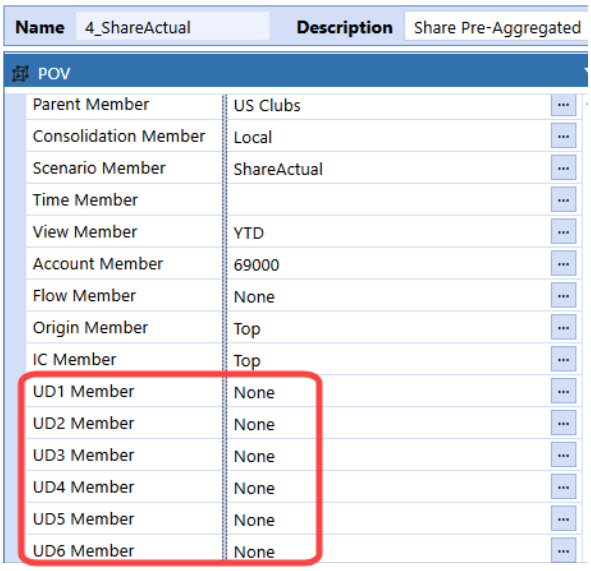
> **NOTE:** You cannot use data unit dimensions in the Member Filters or Member Filters to Exclude properties (entity, time, consolidation, scenario).

Pre-aggregated Members: Determines whether to share or copy data from a parent member, source, to a base member, target.

> **Example:** If you query the top member of a large dimension such as a UD, the aggregated total is calculated on-the-fly each time.

Set the top member to a base member to pre-aggregate if the detail of a dimension is not needed. This alleviates repetitive calculations for the same number. For example:

UD1#Top=UD1#None,UD2#Top=UD2#None,UD3#Top=UD3#None,UD5#Top=UD5#None, UD5#Top=UD5#None, UD6#Top=UD6#None



> **NOTE:** Ensure the base members shown above are included in the Member Filters property and the parents are included in the Member Filters to Exclude property.

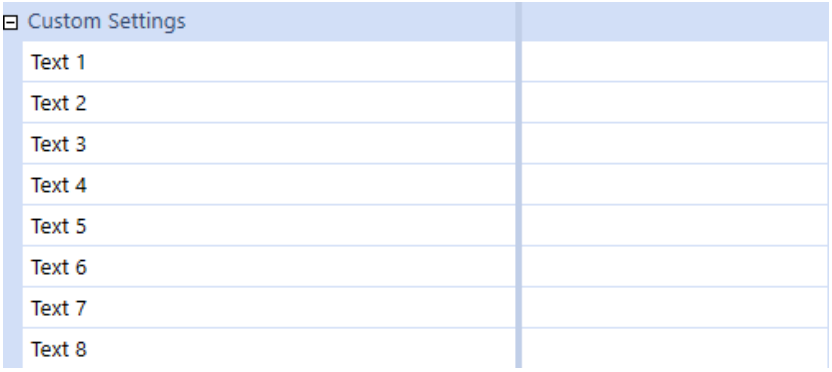
Options: Optional and variable settings that provide additional control when running a hybrid share or copy.

When setting options, remember:

- Options are name-value pairs.
- Ensure the option names, definitions, and syntax are accurate and include any custom name-value pairs if a business rule is used to copy.
- Create a comma separated list if you use multiple options.

Custom Settings

Custom Settings, Text1 -Text 8, enable you to use custom attributes for business rules, member filters, or transformation rules. The value can be changed in the tag over time as the business changes, or by scenario type.



Relationship Properties Tab

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Scenario Dimensions** and select a scenario member. Select the **Members** tab and then the **Relationship Properties** tab.



See the following category descriptions under Configurable Dimensions:

- General

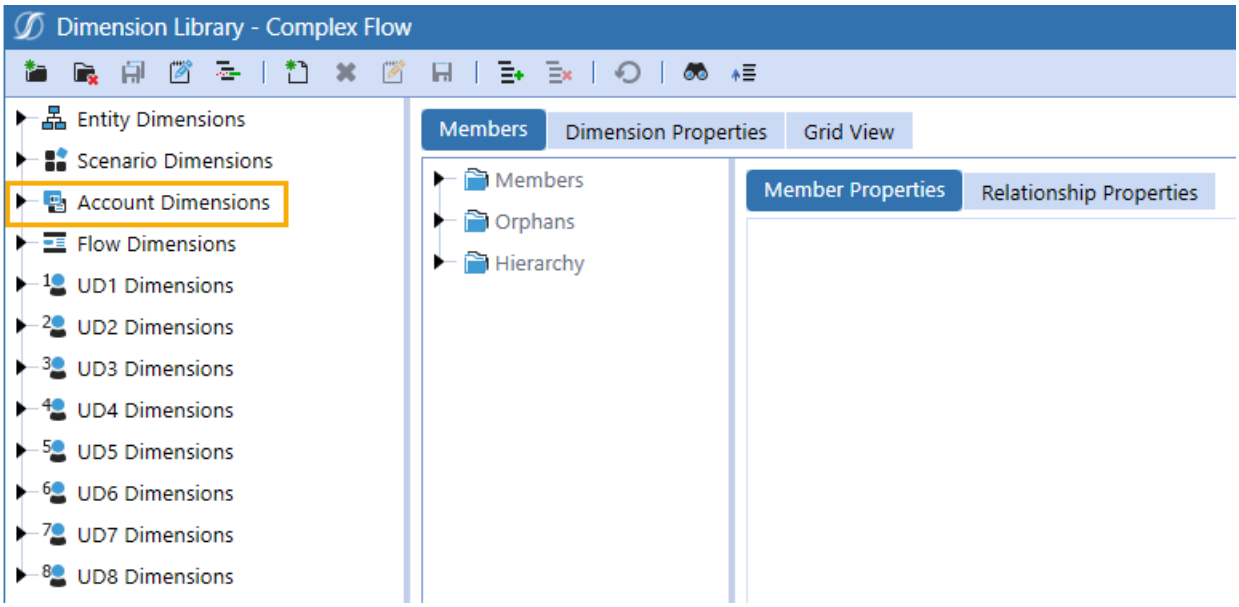
  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Position Within Parent

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

Account Dimension

The account dimension stores both financial and non-financial members.



Account members should be organized hierarchically so data entered for each account will aggregate from a child to a parent member based on account type.

> **Example:** Members with an account type of revenue aggregate as a positive number, and members with an account type of expense aggregate as a negative number in the calculations.

Account hierarchies drive reports, such as income statements and balance sheets.

Member Properties Tab

To view the **Member Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Account Dimensions** and select an account member. Click the **Members** tab and then the **Member Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Descriptions

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Security
- Settings
- Aggregation
- Vary By Cube Type

- Vary By Scenario Type
- Vary By Scenario Type And Time

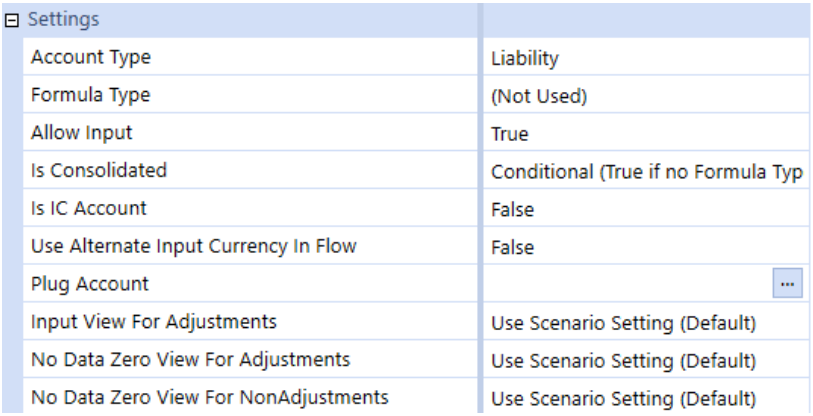
Security

Controls who can view the selected account dimension member.



Display Member Group: Select the ellipsis to set which security group can see that the account member exists within a list of accounts.

Settings



Account Type: Determines how the accounts roll up to a parent. Choose from the following options:

- **Group**: Setting used to make the account organizational-only and will not aggregate data
- **Revenue**: Setting used to tag income accounts with a revenue attribute
- **Expense**: Setting used to tag expense accounts with an expense attribute. The amount entered does not have to be negative because of this attribute.
- **Asset**: Setting used to tag asset accounts
- **Liability**: Setting used to tag liability accounts. The amount entered does not have to be negative because of this attribute.

  > **NOTE:** Equity accounts should set account type equal to liability because an equity account type does not exist in OneStream.

- **Flow**: Setting to hold values that act like an income statement account and have a periodic and year-to-date value. This account does not translate.
- **Balance**: Setting to hold values that act like a balance sheet account that are at a particular time. This account does not translate.
- **BalanceRecurring**: Setting for a balance sheet account that does not change over time such as an opening balance. This account does not translate.
- **NonFinancial**: Setting for informational accounts and not financial such as headcount or square footage. This type is used for legacy purposes such as upgrading from older systems. This account does not translate.

  > **NOTE:** NonFinancial and balance account types are both available for legacy purposes. However, a NonFinancial data cell is not impacted by the flow member's switch type property setting. See Flow Dimension for more information on the switch type setting.

- **DynamicCalc**: An account that calculates on the fly and does not need other formulas to run in order to calculate.

  > **Example:** Ratios that can be calculated as needed.

Formula Type: Determines the behavior related to the member's formula. Choose from the following options:

- **FormulaPass1…FormulaPass16**: This setting controls which rules run, and in what order, to ensure the desired calculation result. Set to Formula Pass X to define when the calculation should run and whether it is dependent on a calculation from other formulas to derive its value. Formulas are included in the account metadata and can be shared between cubes.
- **DynamicCalc**: Set to Dynamic Calc to calculate the member's value every time the cell needs to be displayed without storing the result
- **DynamicCalcTextInput**: Set to DynamicCalcTextInput to run the calculation dynamically and open the data cell for text annotations on a cube view

For more details on using formulas,See "Formulas" in the *Application Tools Guide*.

Allow Input: Setting to determine whether data input is allowed.

- Set to **True** (default) if:
  - Data input for the account member is allowed, or
  - Specific scenarios or entities need input. Then use the cube's Conditional Input settings to control input.
- Set to **False** to make the member read-only. Examples include:

- When the account has a formula and you don't want the formula changed.
- If the account member is a parent-level member. Setting the parent to false removes the member as a possible mapping target, which prevents data from erroneously being mapped to it.

Is Consolidated: Determines if the data for the member will consolidate to its parent.

- Set to **Conditional (True if no Formula Type (default))** to have accounts with a formula type specified included in the calculation and consolidation process. Determines which accounts will be part of the consolidation. If the account has a formula type, the account member will be calculated and consolidated only if the setting is true.
- Set to **True** (regardless of formula type) to include the account in the calculation and consolidation process, irrespective of whether it has an associated formula type.
- Set to **False** if the account should not be consolidated.

> **NOTE:** Increase performance by setting **Is Consolidated** to **False** for each member that does not need to be consolidated.

Is IC Account: Determines if an account is an intercompany account.

- Set to **Conditional (True if Entity not same as IC)** or **True** to have this account identified as an intercompany account and allow transactions to be processed based on the settings in the Vary by Cube Type section under IC Constraints and IC Member Filter.

> **NOTE:** Conditional is a more common usage than True. Setting to conditional will prevent an entity from recording IC with itself as the IC partner. If set to True, an entity can record IC with itself as the partner.

- Set to **True** to have this account identified as an intercompany account and allow transactions to be processed based on the settings in the constraint section under IC Constraints and IC Member Filter.
- Set to **False** (default), to ensure the account will not be identified as an intercompany account.

Use Alternate Input Currency In Flow: Determines if alternate currency should be used.

- Set to **True** to have the account use the historical currency override.
- Set to **False** to ensure the account does not use an alternate currency.

Plug Account: Can be used to select an account as the Intercompany Plug Account to handle any non-eliminating transactions. To select an account:

1. Select the ellipses to open the **Select Member** dialog box.
2. Select the **Dimension** drop-down to select the dimension where the account is located.
3. Use the **Hierarchy** or **Search** tabs to select the account.
4. Select **OK**.

> **NOTE:** Multiple accounts can be reclassified to the same plug account.

Input View for Adjustments / No Data Zero View For Adjustments / No Data ZeroView For NonAdjustments: These settings define how the account member will perform for adjustments and non-adjustments. For each setting:

- The scenario's Input View for No Data Zero View For Adjustments property setting dictates how the account will process as an adjustment.
- The default is **Use Scenario Setting (Default)**.

- Periodic or YTD can be selected to override the scenario's Input View for Adjustments and the No Data Zero View settings.

Aggregation

Aggregation settings enable or disable aggregation for specific dimensions.

Aggregation	
Used On Entity Dimension	True
Used On Consolidation Dimension	True
Enable Flow Aggregation	True
Enable Origin Aggregation	True
Enable IC Aggregation	True
Enable UD1 Aggregation	True
Enable UD2 Aggregation	True
Enable UD3 Aggregation	True
Enable UD4 Aggregation	True
Enable UD5 Aggregation	True
Enable UD6 Aggregation	True
Enable UD7 Aggregation	True
Enable UD8 Aggregation	True

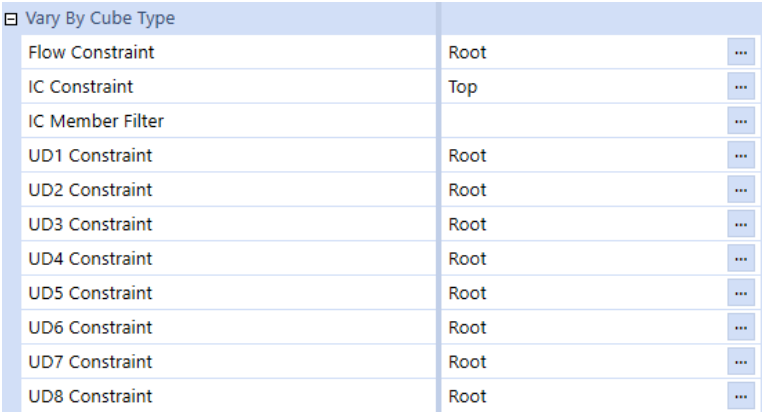
Select a cell to set each property to **True** or **False**. Set to **True** to allow aggregation of the account member in the specified dimension or **False** to turn off aggregation. Dimension members used for informational purposes are typically set to **False**.

- Used On Entity Dimension
- Used On Consolidation Dimension
- Enable Flow Aggregation
- Enable Origin Aggregation
- Enable IC Aggregation
- Enable UD1…UD8 Aggregation

Vary by Cube Type

Properties in the Vary by Cube Type category can be used to set constraints. Constraints dependent on specific dimensions can vary by cube type.

Constraints are used to restrict intersections to which data can be entered for a member and can be set on flow, intercompany, and user defined members.



Vary By Cube Type	
Flow Constraint	Root
IC Constraint	Top
IC Member Filter	
UD1 Constraint	Root
UD2 Constraint	Root
UD3 Constraint	Root
UD4 Constraint	Root
UD5 Constraint	Root
UD6 Constraint	Root
UD7 Constraint	Root
UD8 Constraint	Root

For complete definitions of each property, see Entity Dimension.

Vary by Scenario Type

The Vary by Scenario Type property allows for the selected account member to be used with a Workflow Channel, for a specified scenario type.



Vary By Scenario Type	
Workflow Channel	Standard

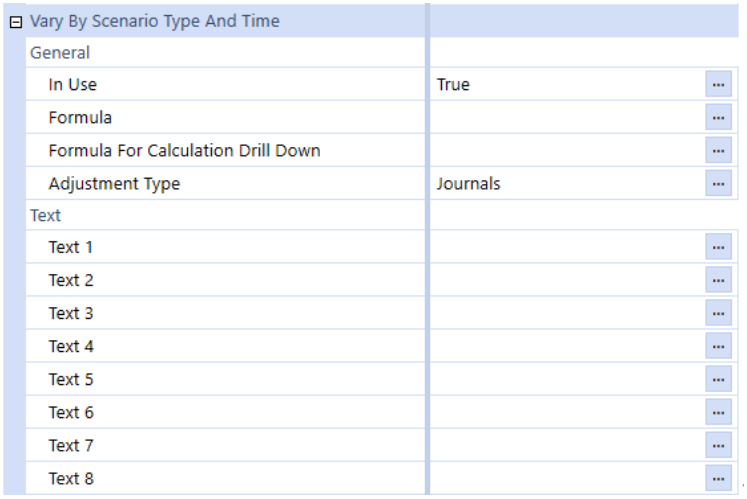
Select the ellipsis to set the **Workflow Channel** to one of the following options:

- **Standard**: A basic member that acts as the default Workflow Channel for account members and Workflow Profile Input Children.
- **NoDataLock**: A special member that only applies to a metadata member (account or UDx) and should not participate in a workflow channel grouping scheme. This is the default value for any UDx Member.

If other Workflow Channels have been created, in addition to these two default channels, they will be available for selection.

Vary by Scenario Type and Time

Accounts can have different attributes based on the scenario type and time, such as associated formulas and the ability to define the method used for adjusting data. The time display for any time varying properties uses the standard time dimension profile. Text properties enable you to tag the selected account.



Select the ellipsis to customize the settings on each of the below properties:

In Use: Keeps historical data available while allowing the ability to close the account without using No Input rules.

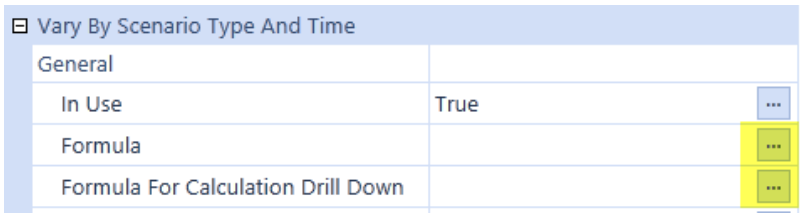
- Set to **True** (default) to have the account in use.
- Set to **False** to turn off the ability to use an account based on time.

Formula: An individual formula kept with the account across cubes which can vary by time.

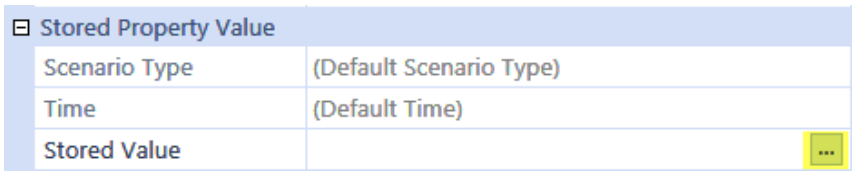
> **Example:** For instance, if a calculation changes over time, but the historical interpretation of that formula needs to be saved. For more information, see the "Formula Guide", in About the Financial Model.

Formula for Calculation Drill Down: Determines if drill down is allowed on members with attached formulas. A specific formula for drill down is required to display the original formula used and the members that activate the drill down back to the original values. To set the **Formula** and **Formula for Calculation Drill Down**:

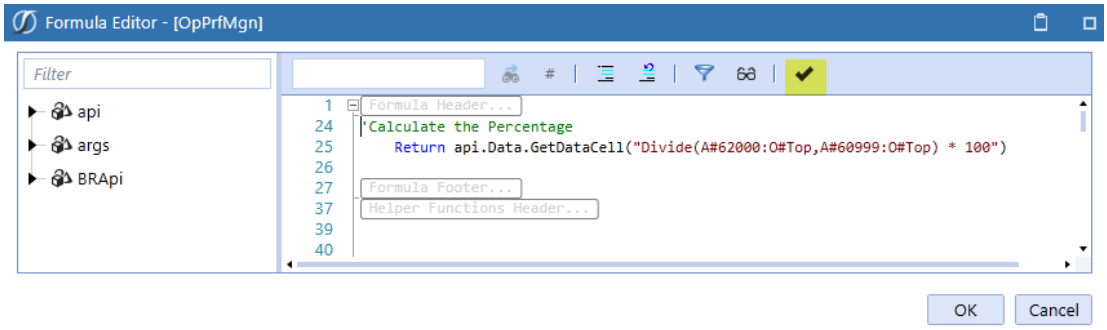
1. Select the ellipsis button located in the **Formula** or **Formula For Calculation Drill Down** fields.



2. Select the ellipsis button in the **Stored Value** field in order to input a formula.



3. Type the individual formula in the **Formula Editor** dialog for the specific account.



4. Select the tick icon to check that the formula was compiled correctly.
5. Select **OK**.

> **TIP:** Formulas can be stacked if a particular account intersection needs to have a special formula different from a normal process. Formulas are not limited to one per metadata member.

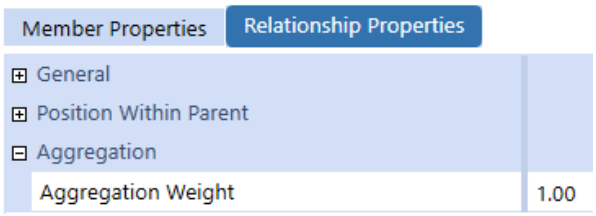
Adjustment Type: Limits the use of adjustments over time. To set this field:

1. Select the ellipsis to open the **Varying Member Property - Adjustment Type** dialog.
2. Select the **Stored Value** cell to generate a drop-down menu.
3. Choose from one of the following options:
    a. **Not Allowed**: Adjustments are not allowed, at any time
    b. **Journals**: Adjustments are allowed for journal entries
    c. **Data Entry**: Adjustments are allowed when entering data manually

Text 1…Text 8: Set custom attributes for use in business rules, member filters, or transformation rules. Change the value over time as business needs change or by scenario type. If multiple attributes are required in a single text field, separate with a comma.

Relationship Properties Tab

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Account Dimensions** and select an account member. Select the **Members** tab and then the **Relationship Properties** tab.



See the following category descriptions underConfigurable Dimensions

- General

> **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Position Within Parent

> **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

Aggregation

Defines the proportion of data for the selected member which will aggregate to the parent.

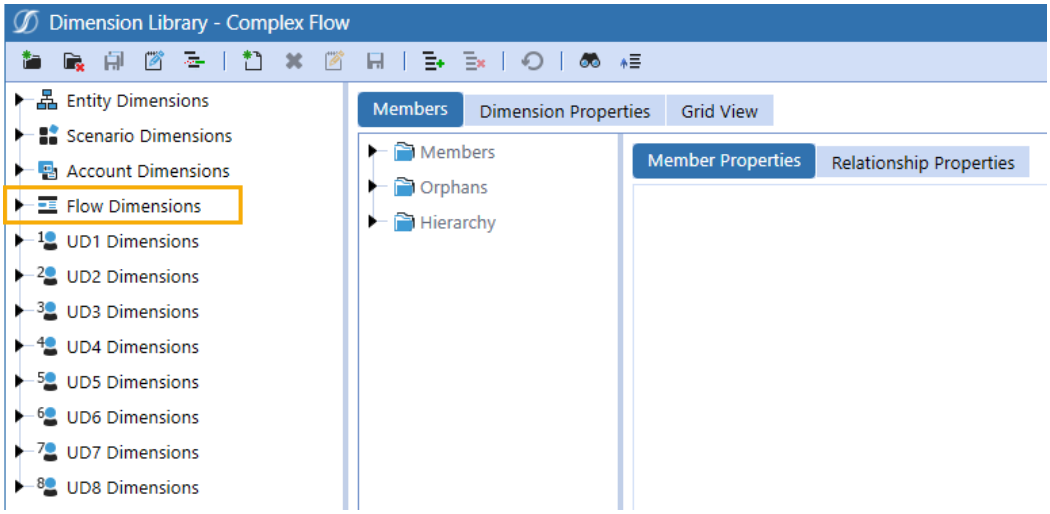
> **Example:** An aggregation weight of 0.50 means that 50% of the data stored against this member will aggregate to its parent.

Aggregation Weight: The Aggregation Weight setting is available in account, flow, and UD Dimensions. Weights on base entities can change based on its parent. Set the weight to 0 to ensure the member does not aggregate.

> **Example:** If a Member is reused in a Dimension, but it does not need to sum up more than once, set the weight to 0.

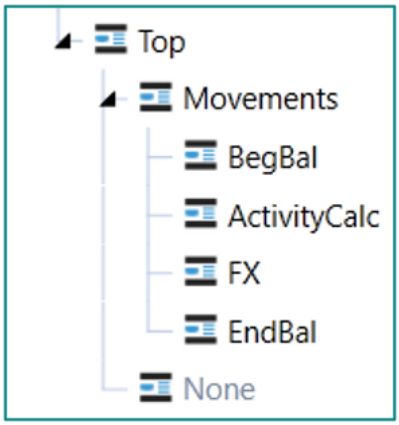
Flow Dimension

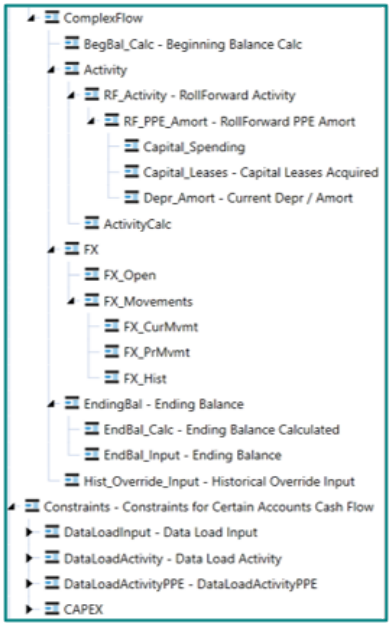
The flow dimension includes members that provide increased visibility into account movements and context on how an account changed from one period to the next.



Members of this dimension aggregate up, and hierarchies can be simple or complex.

Simple Flow **Complex Flow**





Additional settings help with historical currency overrides and eliminate many common custom scripts in order to perform calculations.

Member Properties Tab

To view the **Member Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Flow Dimensions** and select a flow member. Select the **Members** tab and then the **Member Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Descriptions

> **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Security
- Settings
- Flow Processing
- Vary By Scenario Type and Time

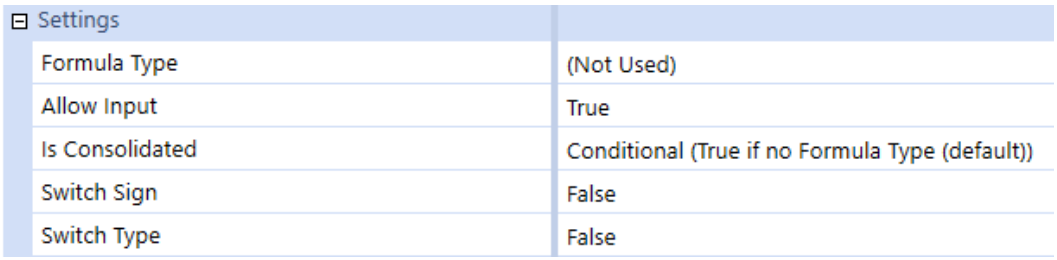
Security

Controls who can view the selected member and any of its children.



Display Member Group: Select the ellipsis to determine which group can see this dimension from the **Application** > **Cube** > **Dimensions** list.

Settings



See Account Dimension for **Formula Type**, **Allow Input**, and **Is Consolidated** fields.

Switch Sign: The flow member is linked to an Account through flow constraints.

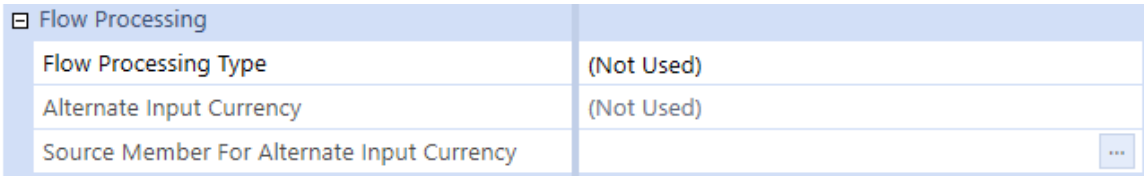
- Set to **True** to switch the sign of data for the flow member.
- Set to **False** to keep the sign as is.

Switch Type: Switches the type of data based on the account attribute, for example setting an asset to a revenue. This is useful when treating roll forward accounts as income statement accounts in the balance sheet.

- Set to **True** to switch the type.
- Set to **False** to keep the type.

Flow Processing

Use these settings when defining an alternate input currency. The flow dimension can hold both values of a dollar override and the settings differ based on how this setting is used. To be used, the account member property field, **Use Alternate Input Currency in Flow**, must be set to **True**, except when you set the **Flow Processing Type** field to **Is Alternate Input Currency for All Accounts**. See "Settings" in Account Dimension.



Flow Processing Type: Select within the cell to generate a drop-down menu with the following options:

- **Is Alternate Input Currency**: Indicates if data should be input into this flow member in a currency other than the entity's local currency. It is used when applications that span multiple currencies are required to override the translated amount with a specific value.

  This setting works similarly to **Is Alternate Input Currency for All Accounts**, except it only applies to a specified list of accounts that are configured in the account dimension.

  If selected, this setting activates the **Alternate Input Currency** property where you can select a specific currency from the list configured through the application properties.

  To override this setting on individual accounts, set the **Use Alternate Input Currency** property to **True** on the account member.

- **Is Alternate Input Currency for All Accounts**: Indicates if data should be input into this flow member in a currency other than the entity's local currency for all accounts. Since this opens currency overrides for all accounts, use cases will vary.

  If selected, this setting activates **Alternate Input Currency** property, where you can select a specific currency from the list configured through the application properties.

  > **NOTE:** If used, the account member property **Use Alternate Currency in Flow** does not need to be set to **True**.

- **Translate using Alternate Input Currency, Input Local**: Overrides the translated value with the amount input at the local currency level. If selected, this setting activates the **Source Member for Alternate Input Currency** property where you can select the specific flow member to override the value for the current flow member.

- **Translate using Alternate Input Currency, Derive Local**: Overrides the translated value and derives the local currency value based on the local currency rate. If selected, this setting activates **Source Member for Alternate Input Currency**. This setting should not be used in a trial balance unless accounting for the possible out of balance condition.
- **Alternate Input Currency**: Lists available currencies that determine the source value override currency. This typically matches the default currency of the application.

  > **Example:** If a USD override is required, set the property to USD. If a EUR is override is required, set the property to EUR.

- **Source Member for Alternate Input Currency**: Defines the specific flow member to override the value for the current flow member.

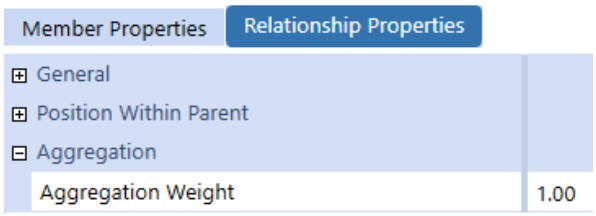
Vary by Scenario Type and Time

See Account Dimension.

Vary By Scenario Type And Time	
General	
In Use	True
Formula	
Formula For Calculation Drill Down	
Text	
Text 1	
Text 2	
Text 3	
Text 4	
Text 5	
Text 6	
Text 7	
Text 8	

Relationship Properties Tab

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Flow Dimensions** and select a flow member. Select the **Members** tab and then the **Relationship Properties** tab.



See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Position Within Parent

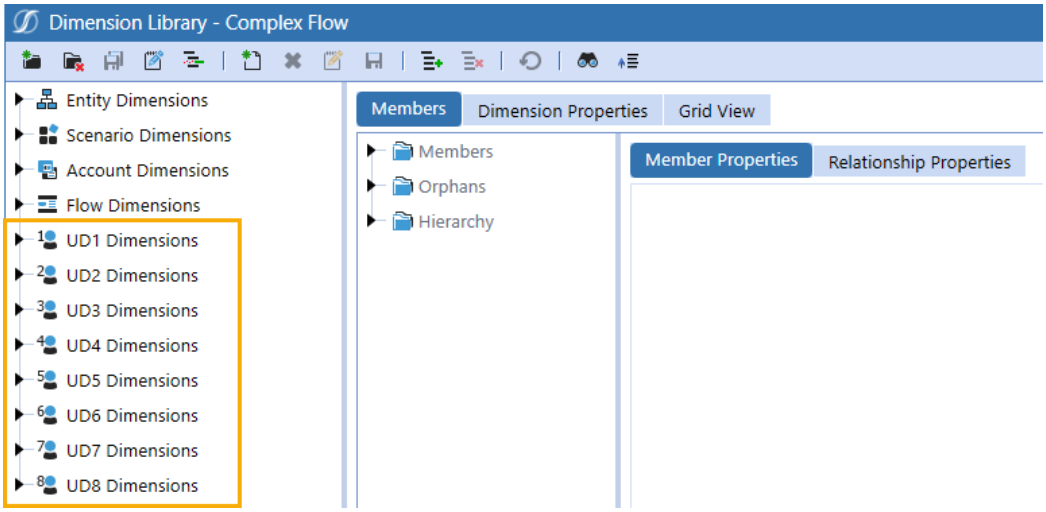
  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

Aggregation Weight

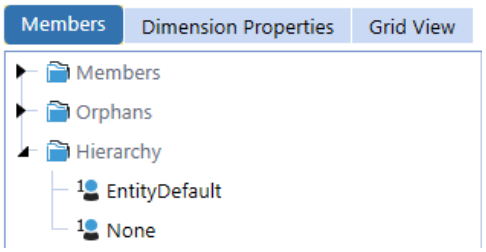
See "Aggregation" in Account Dimension

User Defined Dimensions 1–8

Take full advantage of Extensible Dimensionality for inheriting and extending across business units and scenario types with user defined dimensions. There are eight user defined dimensions, but not all eight need to be used. When determining the order in which to use them, it is recommended to define the larger or more significant dimensions first using UD1 or UD2. Stacking hierarchies in one user defined dimension does not affect calculation time, because performance is primarily based on the number of stored data cells rather than the number of possible intersections.

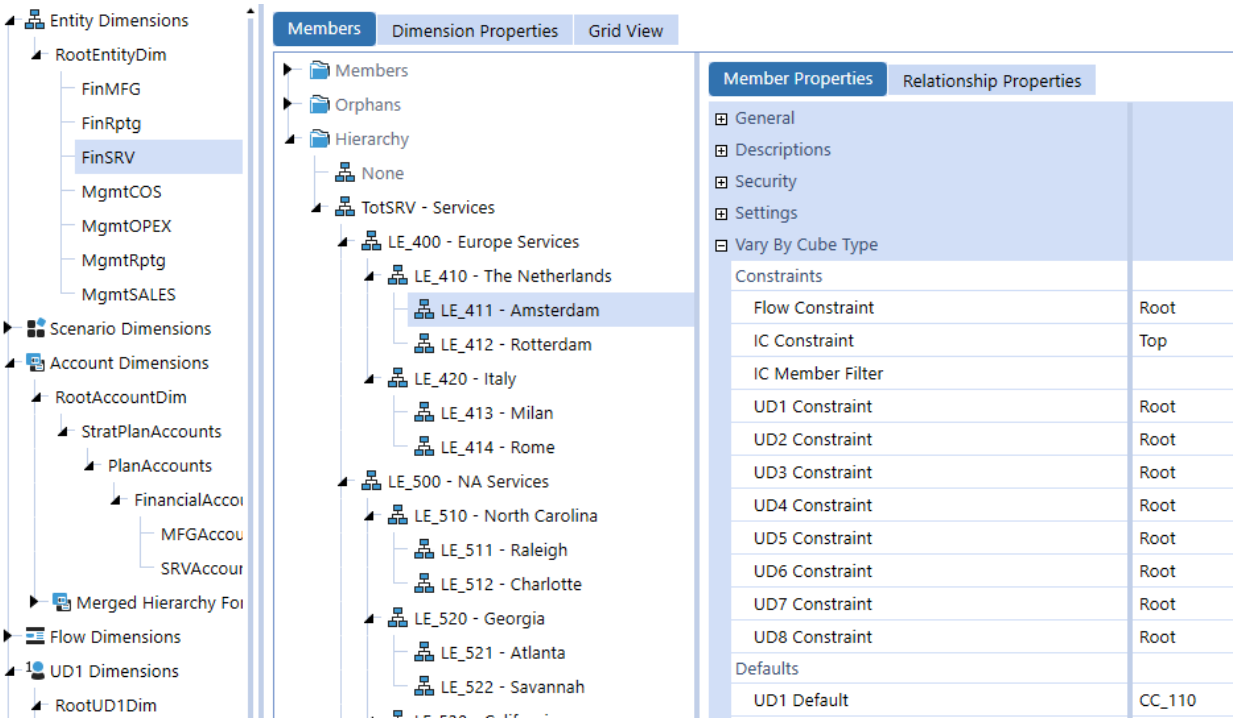


Each user defined dimension has an **EntityDefault** member that is used to assign attributes to an entity. This is set on the specific entity in the Vary By Cube Type settings for each user defined dimension. With this feature, an entity can have a specific default, which reduces the need to map every entity to a common tag, such as region or division, for data import and form-based data entry. You can select the **EntityDefault** member without knowing the specific entity setting for each of these user defined dimensions. However, using this setting impacts consolidation time, because more intersections will be in the financial model.

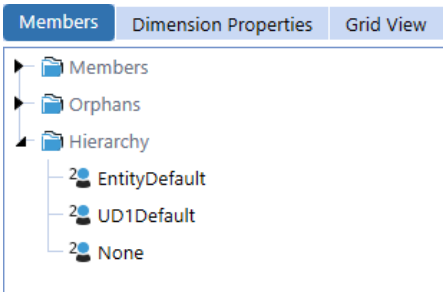


A UD1 through UD8 default membership can be defined to a user defined dimension, such as a cost center.

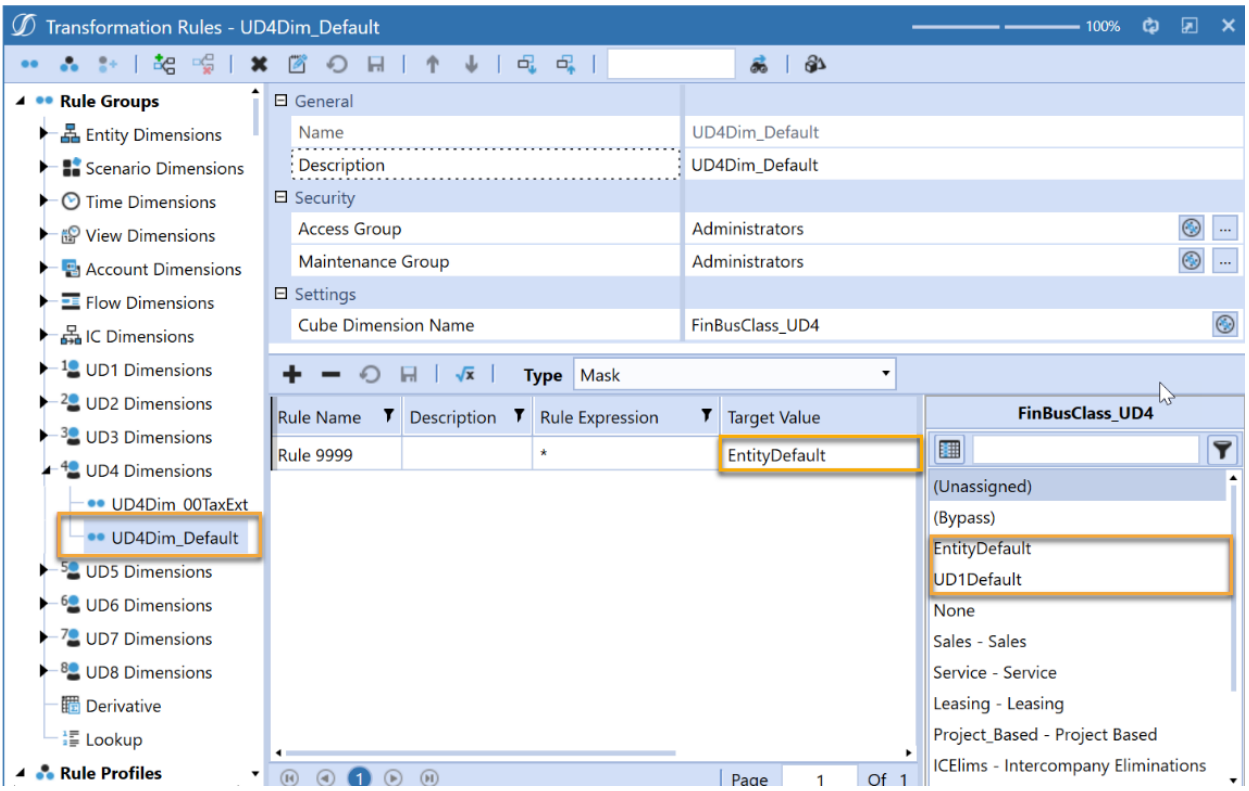
> **Example:** This entity falls in the Europe Services region and every import, form edit, and journal entry should classify this entity as being associated with the CC_110 - Sales cost center. The entity never has to explicitly map to the cost center. When data is loaded, it is directed to the EntityDefault member and the user defined setting will be applied automatically. This can have a negative impact on the consolidation if not used in a user defined dimension with a limited number of members.



UD2 through UD8 also have a **UD1Default** member. **UD1Default** can be used for constraints and in transformation rules.



In the following example, the EntityDefault member was used in UD4 mappings and the same could be used for UD1Default.



Member Properties Tab

To view the **Member Properties** tab, go to **Application** > **Cube** > **Dimensions** and select a user defined dimension member. Click the **Members** tab and then the **Member Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Descriptions

> **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Security

> **NOTE:** The field **Display Member Group** is in the entity dimension.

- Settings

> **NOTE:** The fields **Formula Type** and **Allow Input** are in the account dimension. See Account Dimension.

- Vary By Cube Type
- Vary By Scenario Type
- Vary By Scenario Type and Time

Settings

Is Consolidated

- **Conditional (True if no Formula Type and no Attribute (default))**: The data from the children of this entity is consolidated. This entity will equal the total of its children.
- **True (regardless of Formula Type and Attribute)**: The results of the dimension and the attribute are consolidated. Consolidate or aggregate user defined attributes when they reference entity, as the reference dimension to view results at the parent entity instead of having the parent entity use the same algorithm as the base entities.
- **False**: The data will not be consolidated. Set to False when using the parent entity strictly for grouping purposes. Also, setting Is Consolidated to False helps with consolidation performance times, because the consolidation will not be performed at the parent entity.

Alternate Currency for Display: Use this setting to change the cube view grid currency. It does not recalculate the member based on the currency. This requires a formula on the member to recast the transaction from another member to the current currency.

Is Attribute Member: Set to **True** to enable the user defined member as an attribute. See User Defined Members as Attribute Members and Use Attributes in Business Rules.

Source Member For Data: If Is Attribute Member is set to True, this represents the member in the user defined dimension containing the current attribute that will be used to define the data returned. This member must be a base level member, such as the None member or a calculated base member. A hierarchy parent member cannot be used.

Expression Type: Used for one or two related dimensions and the conditional relationship to return results.

- **Comparison 1 Only**: This is used for an attribute with a single related dimension.
- **Comparison 1 and Comparison 2**: This is used for two related dimensions in which the results will be bound by meeting both related dimension conditions.
- **Comparison 1 Or Comparison 2**: This is used for two related dimensions in which the results will need to meet one of the related dimension conditions.

Related Dimension Type: Identify the dimension to use as a source to be evaluated. To support on-the-fly aggregations, an attribute can reference any account, flow, or user defined dimension. Entity and scenario are stored members. Therefore, attributes will only be reflected as base members. They cannot reference the user defined dimension in which it is contained.

Source dimensions for attributes, which is the assigned Related Dimension Type, are evaluated only on the base member, not parent members. Therefore, the Property, Related Property, and Comparison Text are designed against collecting data at the base level member on the source dimension. When referencing the Text 1 through Text 8 fields, the Vary by Scenario Type and Time can be used. See Vary by Scenario Type and Time.

Related Property: The related dimension can be evaluated on its Name, Description, Name or Description, Name and Description, or Text 1 through Text 8 fields. The related property will be evaluated only on the base members of the related dimension type.

Comparison Text: The text condition being evaluated against the defined related property as the name, description, or text fields. When referencing the text field, the source text field can vary by scenario and time. When referencing the description field, only the default description can be referenced.

Comparison Operator: Sets the evaluation method to compare the related property to the comparison text. This can be done explicitly with = or <> as well as dynamically using the Starts With, Ends With, Contains, or Does Not Contain operators.

After the attribute members are defined and any required comparison text is applied to the source related dimension type, the data will render dynamically. Neither consolidation nor calculation is required to render the results. Any change applied to the source definition of the comparison text, such as a change to a Text property, will immediately be reflected in the results on the attribute members. Similarly, modifying the setting properties of the attribute member will immediately change the attribute results.

Vary By Cube Type

This section is only available for UD1 because you can set a default value for the members of UD2 through UD8. For example, if the UD1 dimension is Cost Centers, you can indicate which UD2 member (such as Department or Product) will be chosen based on the UD1 selection.

UD2 Constraint through **UD8 Constraint**: A constraint is a setting that allows only certain members to be used. If a member is outside the UD constraints applied on the account dimension, an intersection in a cube view will show as red, indicating an intersection that is not valid, and any numbers in that intersection will not aggregate. Constraints applied on the entity and UD1 dimensions will create a green, no input data intersection.

UD2 Default through **UD8 Default**: This is the standard default member that can be mapped for a setting rather than mapping to each member. If the setting is **Default**, the mapping will always go to that default.

Relationship Properties Tab

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** and select a user defined dimension member. Click the **Members** tab and then the **Relationship Properties** tab.

See the following category descriptions:

- General

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Position Within Parent

  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Aggregation

User Defined Members as Attribute Members

A user defined member can be defined as an attribute member by enabling the Is Attribute Member property. When enabled as an attribute, the user defined member becomes a read only member in Cube Views based on the settings for the related reference properties. Data intersections are not loaded directly to an attribute. When setting up a UD Attribute, set the property "Allow Input" to False to prevent inadvertent use in data integration. Its results are derived from references to properties of other members, such as name, description, or text fields. These members act like other dimension members by deriving their values based on the references to other properties, which can then be used in reporting.

The account, flow, and user defined dimensions, as related dimension types, support the attribute members for calc-on-the-fly aggregations at parent members. The dynamically generated results within the attribute members will be automatically aggregated to the parent members.

The UD attribute can consolidate to a parent when a UD attribute's "Related Dimension Type" is Entity. At a Base Level Entity, a UD attribute by Entity will calculate on the fly, and will not add to the data unit. The User Defined Attributes can also be applied to Stored Dimensions such as entity and scenario, but the results will not automatically summarized, or consolidated to parent members. If the results are required at the parent levels, a consolidation or aggregation is required. The **IsConsolidated** property on the UD attribute must be set to True and the results at the parent will be stored values and add to the Parent Entity Data Units.

The use of user defined attribute members can impact application performance, particularly with respect to consolidation time. The impact is caused by the dynamic generation of user defined attribute member data intersections adding to the size of the final data unit. Therefore, the potential intersections derived from user defined attribute members should be included in application data unit analysis. As a guideline, typical application designs should consider the performance evaluation of user defined attribute members if the number of user defined attribute members approaches approximately 2000 items.

The user defined member as an attribute is unique:

- The data will dynamically calculate across each dimension hierarchy.
- The attribute members will not impact the size of the data unit in consolidation. Except when the UD Attribute is related to an Entity Dimension and consolidated or aggregated at a parent entity
- Values derived by attribute members can be referenced by business rules and by member formulas.
- The members are treated as standard dimensions and records in that they will be processed within a cube view in the Allow Sparse Suppression routine supporting large sparse application model reporting.
- The results can be modified by modifying the definition of the reference on the attribute member or from a change on the properties of source members even if those properties vary by scenario type or time.

The model design and use of attributes should consider if the feature is appropriate for the application model. Here are some considerations:

- Attributes may not be appropriate in situations where reporting on the attribute member must be maintained with a high level of data integrity. This is due to the dynamic nature of the attribute where its results are based on properties of other members.
- Attribute results cannot be locked for data integrity. Although the underlying data being referenced will be locked, modifying the definition of the attribute or a change on the properties of the referenced source member may impact the results.
- Since attribute members do not store data, they can be deleted and are not subject to data integrity restrictions if in use. Therefore, dynamic designs of reports and use in rules should be considered.
- Attributes cannot be input or contain formulas. However, they can reference other input members or calculated members as a source.

- Attribute data cannot be extracted to a data file.
- Drill-down based on the attribute member intersection cannot be used to drill back to the stage load results. The source member for Data reference member defined on the attribute must be used.

Use Attributes in Business Rules

Attributes can only be called through a business rule using a DataBuffer. The function property includeUDAttributeMembersWhenUsingAll can be enabled to allow rules to reference the attribute results for use in formulas.

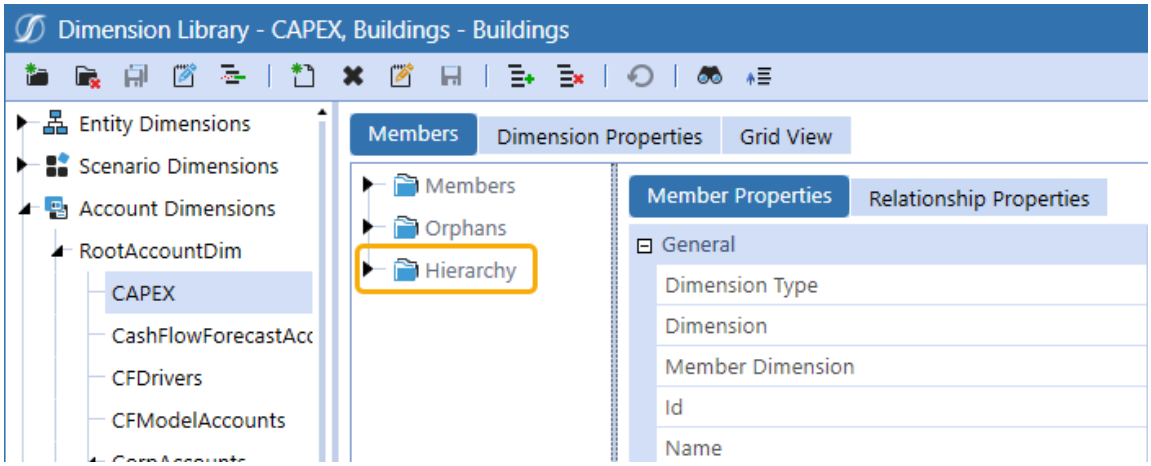
```
Dim objDataBuffer As DataBuffer =
                          api.Data.GetDataBuffer(scriptMethodType,
                           sourceDataBufferScript,
               changeIdsToCommonIfNotUsingAll, includeUDAttributeMembersWhenUsingAll,
expressionDestinationInfo)
```

Parent Dimension

The parent dimension type is the immediate next member up in a hierarchy where a relationship to the member exists.

> **Example:** In the Account dimension, a member called Total Expenses may be a parent of the member Operating.

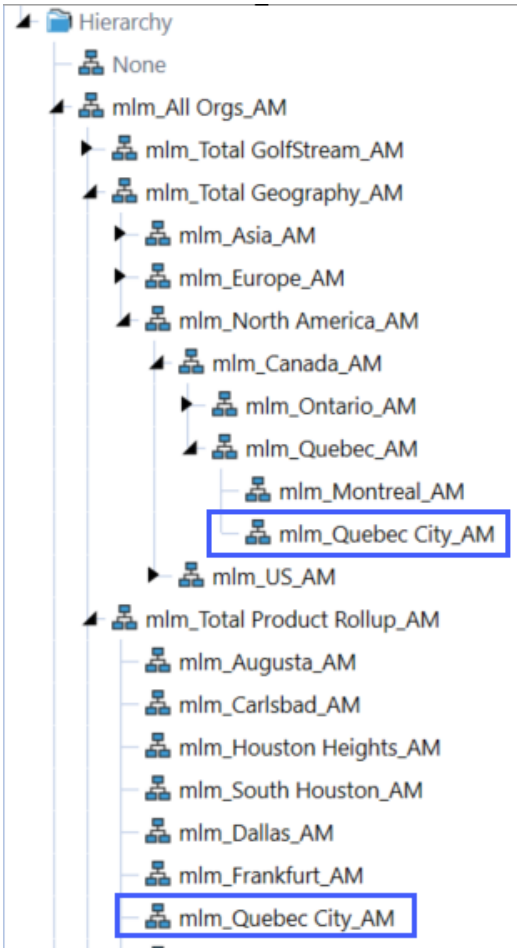
The parent dimension member is defined by the relationships within an entity dimension member hierarchy. To view the hierarchy, go to **Application** > **Cube** > **Dimensions** and select a dimension member. Select the **Members** tab and then select the arrow next to **Hierarchy** to expand the member hierarchy.



Alternate Hierarchies

The parent dimension member must be specified for a data cell value because a member can be part of multiple hierarchies. This is also known as alternate hierarchies. As a result, a single member can have different parent-level members.

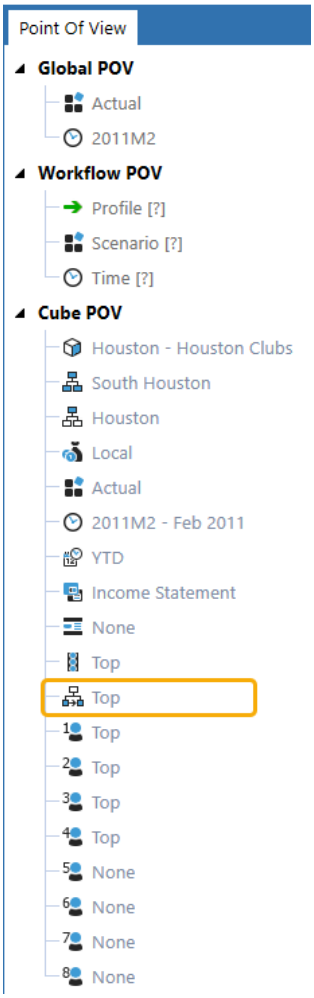
> **Example:** Below, mlm_Quebec City_AM has two different parent-level entity members, as it is both a child member of mlm_Quebec_AM and mlm_Total Product Rollup_AM.

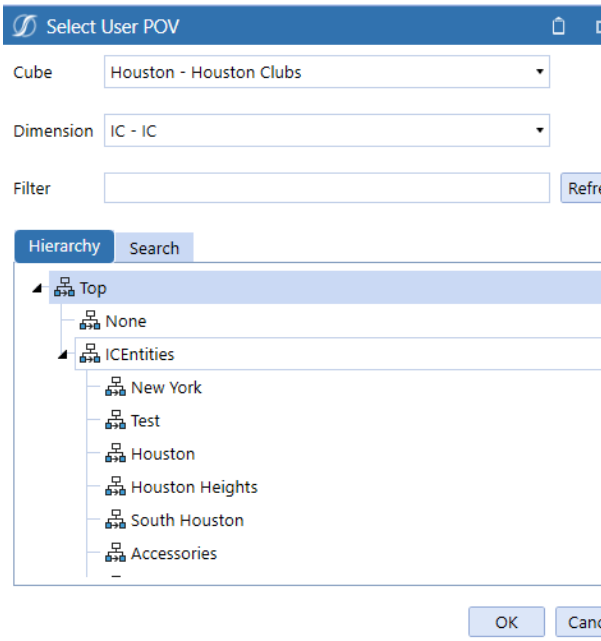


> **NOTE:** The parent dimension does not have a hierarchy of its own as it varies based on the dimension and member selected.

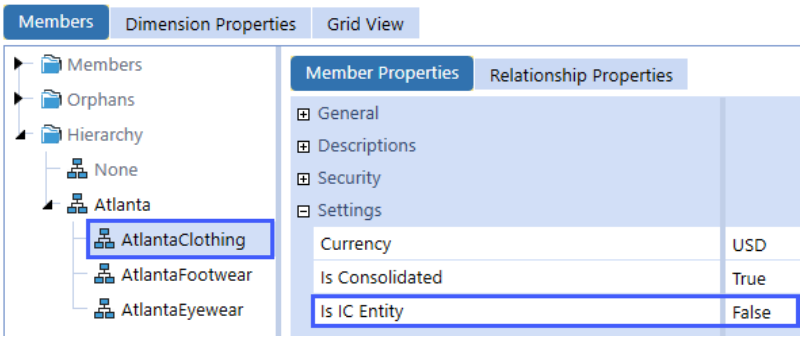
Intercompany (IC) Dimension

The intercompany (IC) dimension specifies the partner entity for an intercompany activity. You can view the IC dimension and its members using the Point of View Pane.





The IC dimension is populated using the **Is IC Entity** property on entity dimension members. To view the **Is IC Entity** property, go to **Application** > **Cube** > **Dimensions** > **Entity Dimensions** and select an entity member. Select the **Members** tab and then the **Member Properties** tab. Expand the **Settings** category to view the **Is IC Entity** property.



When **Is IC Entity** is set to **True**, then an intercompany dimension will be populated and shown in the IC dimension hierarchy. This is only required on base-level entities where the intercompany intersections roll up and eliminate.

> **NOTE:** There are no parent-child relationships in the IC dimension.

Both entities must have **Is IC Entity** set to **True** to have the ability to record a transaction with each other. This protects the integrity of the intercompany data, ensuring that an intercompany account cannot have an intercompany transaction with an entity if it cannot receive an intercompany transaction.

See Consolidation.

Time Dimension

The time dimension includes year and period and determines whether data is stored at a weekly, monthly, or 12/13 period frequency. The number of months and weeks in a quarter can also be set.

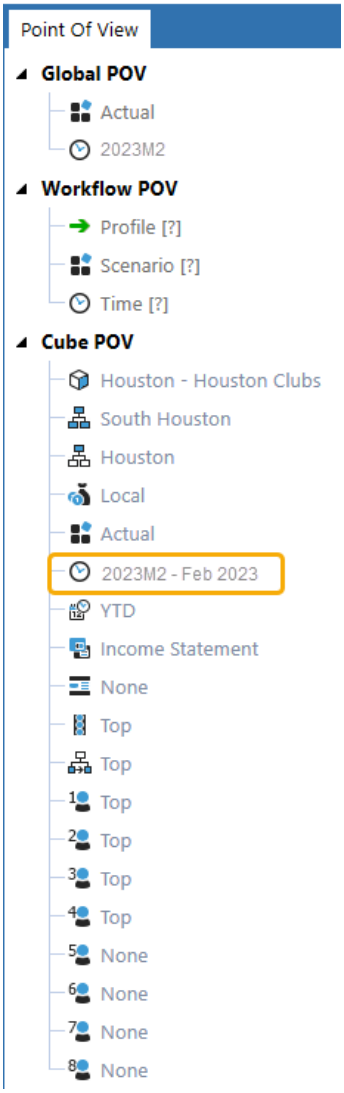
> **Example:** The 445 calendar divides a year into four quarters of 13 weeks, each grouped into two 4-week months and one 5-week month. The longer month may be set as the first (5-4-4), second (4-5-4), or third (4-4-5) unit. This drives the time hierarchy that displays in the point of view.

> **NOTE:** Weeks will only appear in applications setup as weekly.

When considering the time dimension, remember:

- Time dimension members are customized before an application is created and cannot be altered after.
- A new application must be created if a different time dimension is needed later.
- There are separate data tables for each year.
- The time dimension is not viewable in the dimension library because it is not editable. You

can view the dimension using the Point of View pane.



See Time Dimensions in System Tools.

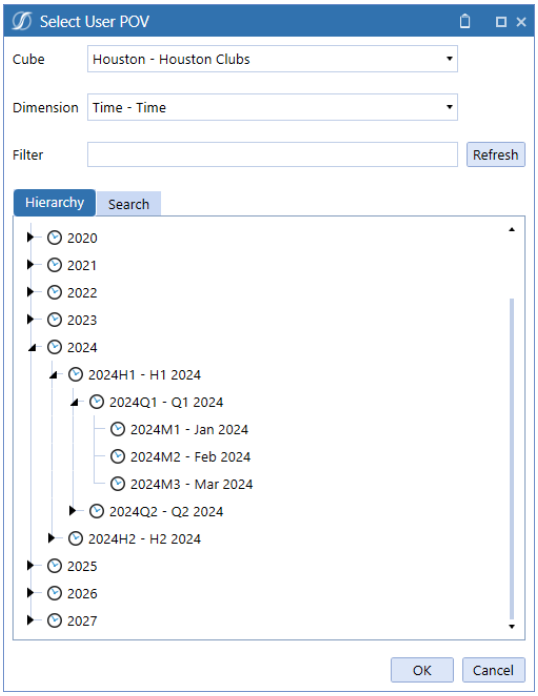
Time Dimension Hierarchy

Members are organized in a way where the descendants of a year include members for periods, including:

- Half-years
- Quarters
- Months
- Weeks

The time hierarchy is not tied to a calendar year. It can be configured to fiscal year by using the fiscal year and Month1 (M1), Month2 (M2), and so on to designate the time frame.

Monthly Example



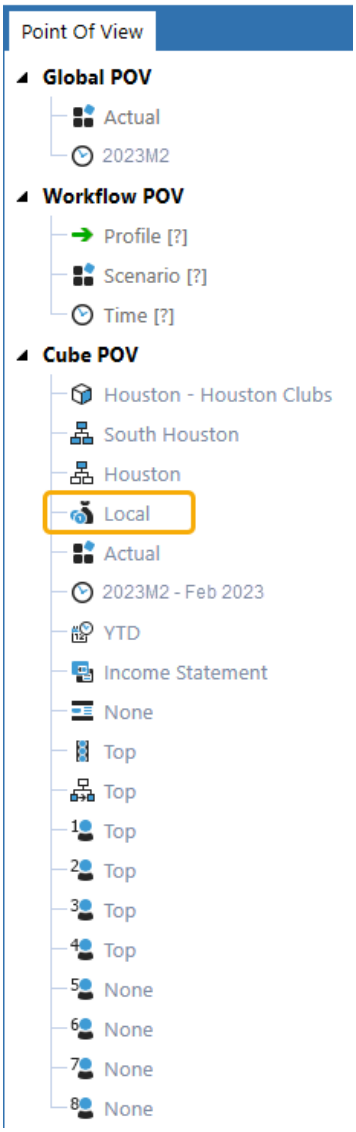
Weekly Example



> **NOTE:** Corporations that do calendar reporting and fiscal year reporting in the same application can run into issues with rolling retained earnings and beginning balances. For example, if a large corporation had two companies using different reporting years, the books must be separate because each company uses a different method. In these instance, two applications, one for each reporting year, would need to be used.

Consolidation Dimension

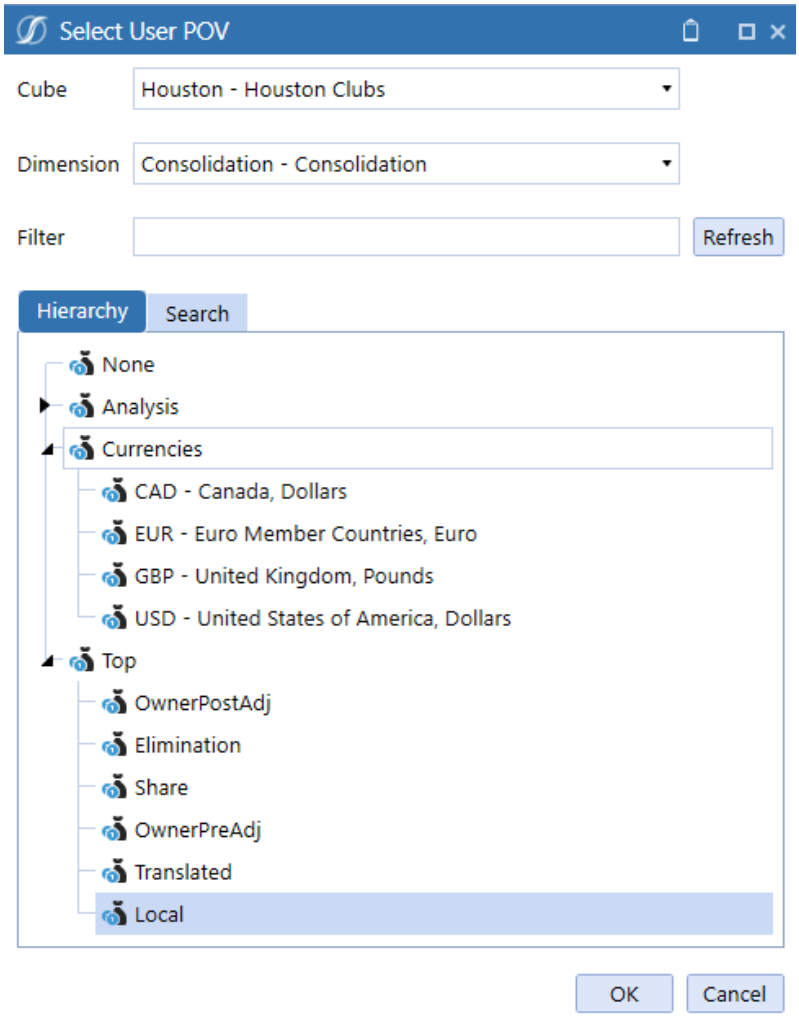
The consolidation dimension member hierarchy shows how entity data rolls up from the local currency to the final numbers that contribute to its parent entity member. This dimension is not editable through the Dimension Library but is viewable using the Point of View pane.



The consolidation dimension shows the translation, share, elimination, adjustment (Journal), consolidation detail, and the currencies assigned to an application. Additional currencies can be added through Application Properties.

The main groups of members in the hierarchy include:

- **Analysis**: Includes an aggregated member for use with processes that do not require a strict statutory consolidation process.
- **Currencies**: Includes one member for each of the currencies assigned to an application in Application Properties.
- **Top**: Used for statutory consolidation, such as a financial close process, where data from child-level entity members are consolidated to parent-level entity members.



See About the Financial Model.

System-defined Dimensions

System-defined dimension types include origin and view. They are not editable.

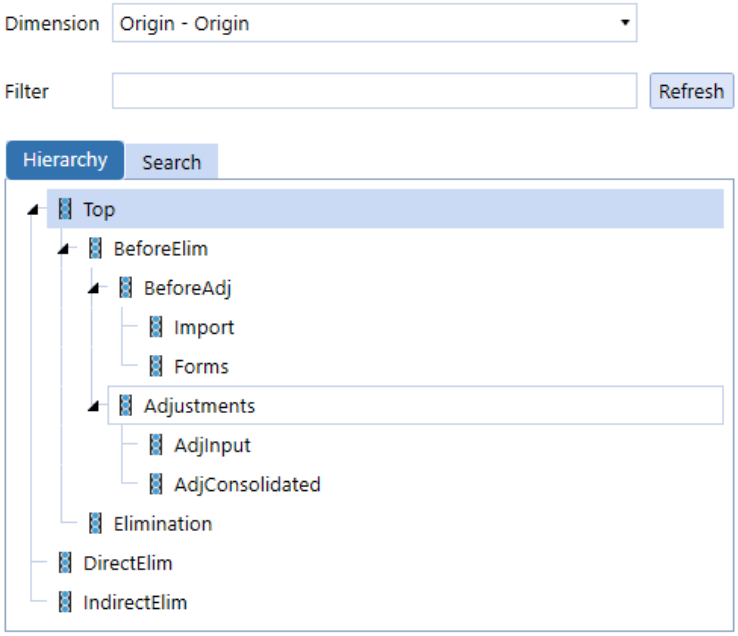
Origin Dimension

The origin dimension shows where data originated from, including, imports, form entries, or adjustments to journals. It enables you to differentiate where the data come from and drill down into data.

This dimension also shows data before and after eliminations which occur at common parent-level entity members between entities' intercompany transactions. See Consolidation.

Origin Dimension Member Hierarchy

Every application has the same origin dimension member hierarchy which includes the following members:

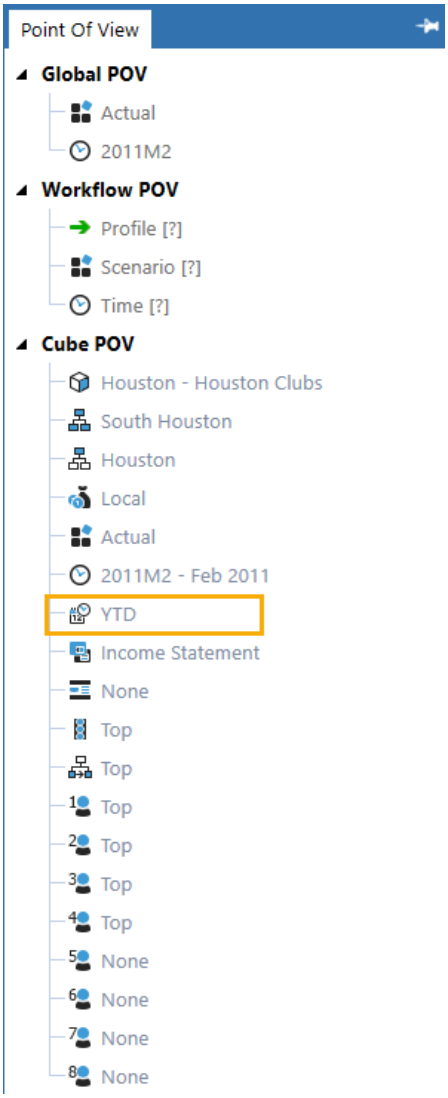


- **Top**: Shows data consolidated after eliminations
- **BeforeElim**: Shows data consolidated before eliminations
- **BeforeAdj**: Shows data before adjustments
- **Import**: Populated through the data loaded during an import workflow, which enables the application to bring in data from an external source
- **Forms**: Populated directly from data loaded into a cube through a form
- **Adjustments**: Shows adjustments made in both the parent-level and child-level entities
- **AdjInput**: Populated from adjustments created through journal entries or forms
- **AdjConsolidated**: Populated when a consolidation is run. The entries in child-level entities are consolidated into this member in the parent-level entity.
- **Elimination**: Populated through the consolidation elimination member

- **DirectElim**: Informational only, it displays eliminations at the first common parent entity. It does not roll up to the top member.
- **IndirectElim**: Informational only, it displays eliminations that have already occurred at a lower level in the entity hierarchy

View Dimension

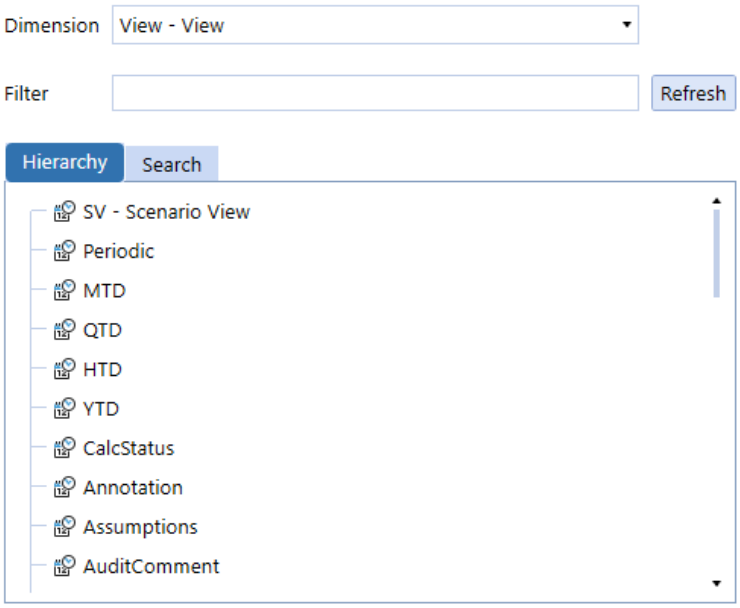
The view dimension is a system-defined dimension and is not accessible in the dimension library. Access the view the dimension using the POV pane.



This dimension is used to view data or text entered into certain dimension members. It allows you to analyze data from different perspectives. All data is stored as YTD by default, but can be viewed and retrieved dynamically using the included common calculations, such as month-to-date (MTD) and quarter-to-date (QTD). Included calculations save you from having to create custom formulas.

Also included is a CalcStatus view, which shows the calculation status for a data unit, and several views into different types of data attachment comments, such as annotation, assumptions, footnote and VarianceExplanation.

The view dimension hierarchy includes members that are the same in every application and there are no parent-child relationships.



Entity Aggregation

Entity aggregation provides the speed and flexibility required for budgeting, planning, and forecasting. Unlike consolidation processes, entity aggregation is simpler and faster because it does not roll up financial data - heavily driven by financial and accounting rules - to a parent level for reporting.

Consider this: "Consolidations are usually crafted to satisfy internal management and external regulatory agency reporting requirements. The most common, effective way to understand the core requirements of a consolidation system is to begin with the end in mind and look at the reports produced by the legacy (or current) system. These usually involve an income statement (Profit & Loss), a balance sheet, and a cash flow statement" – OneStream Architect Factory

How Entity Aggregation Works

Entity aggregation bypasses most statutory financial and accounting rules to quickly consolidate/aggregate data the entity dimension for fast, what if scenario modeling. Entity aggregation uses the entity and the consolidation dimension (aggregated member) to aggregate data with minimal rules.

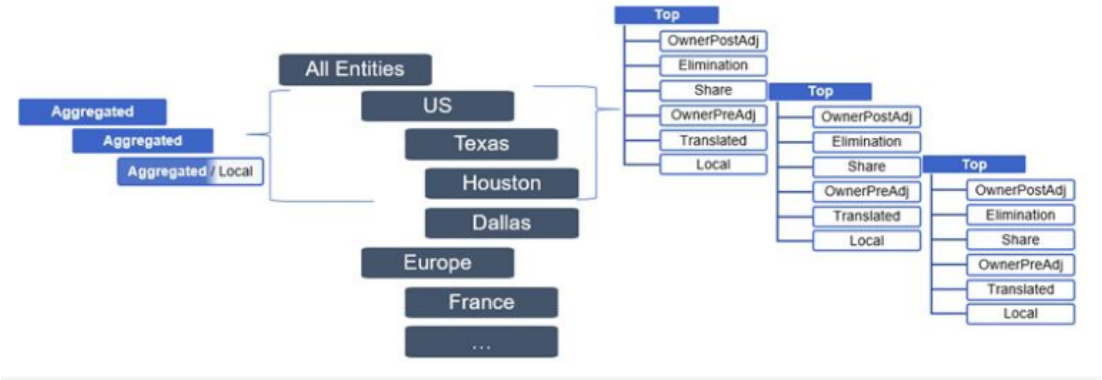
For base level entities, the aggregated member displays the data that is stored in the Local member.

For parent entities, the aggregated member stores the results of the entity aggregation process that occurs on its children.

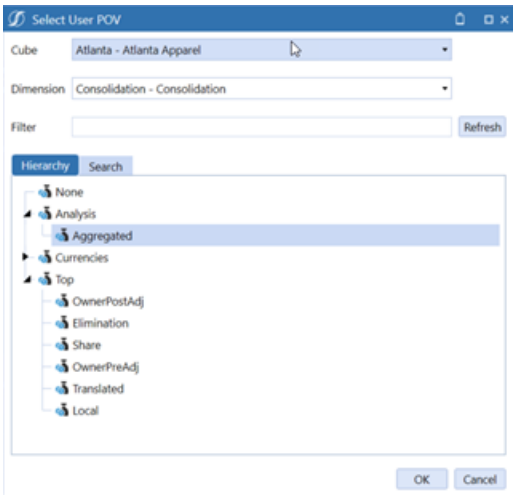
Aggregation Algorithm:

- Execute chart logic (business rules and member formulas) on the local consolidation member for all base entities.
- Execute these steps recursively for each parent and its direct children, from lower-level entities to the parent entities:
    - For each child:
        - Translate stored data in memory.
        - Calculate the share amount in memory
    - Add the data cells from each child in memory.
    - Store the results in the aggregated member for the parent entity.

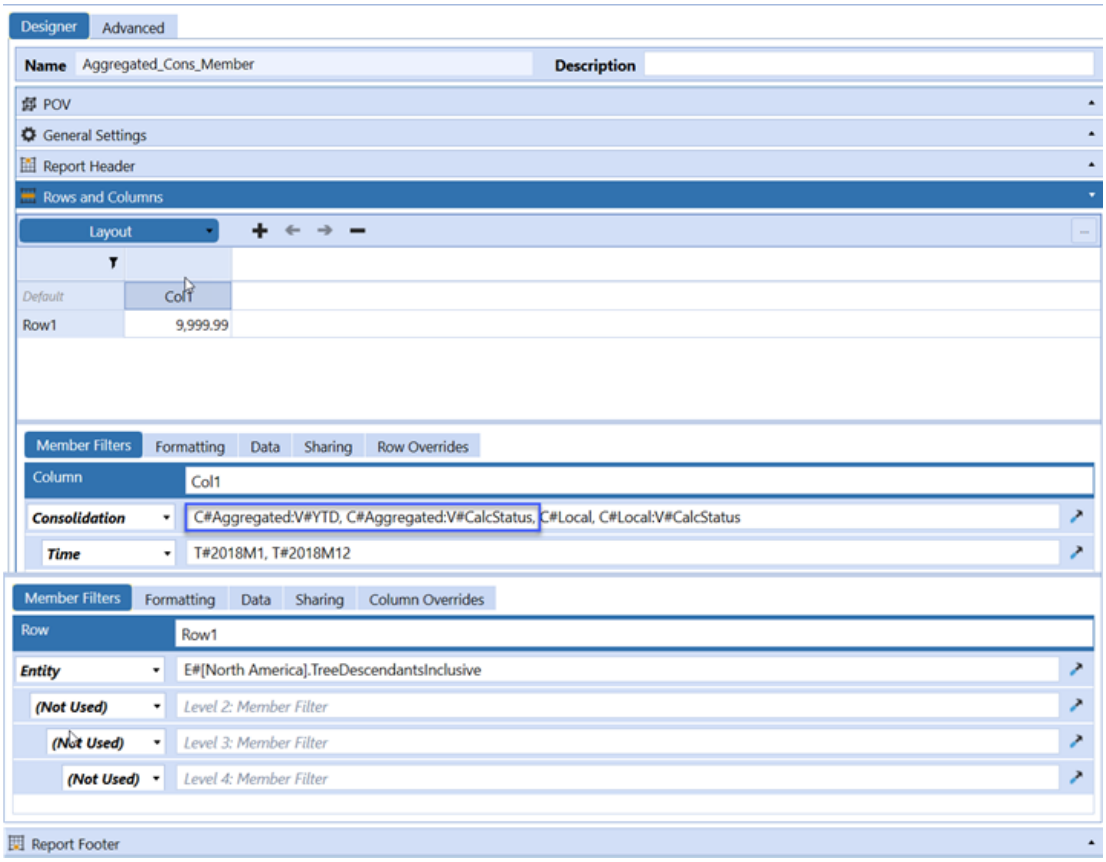
Example of an Entity Aggregation



Aggregated Dimension

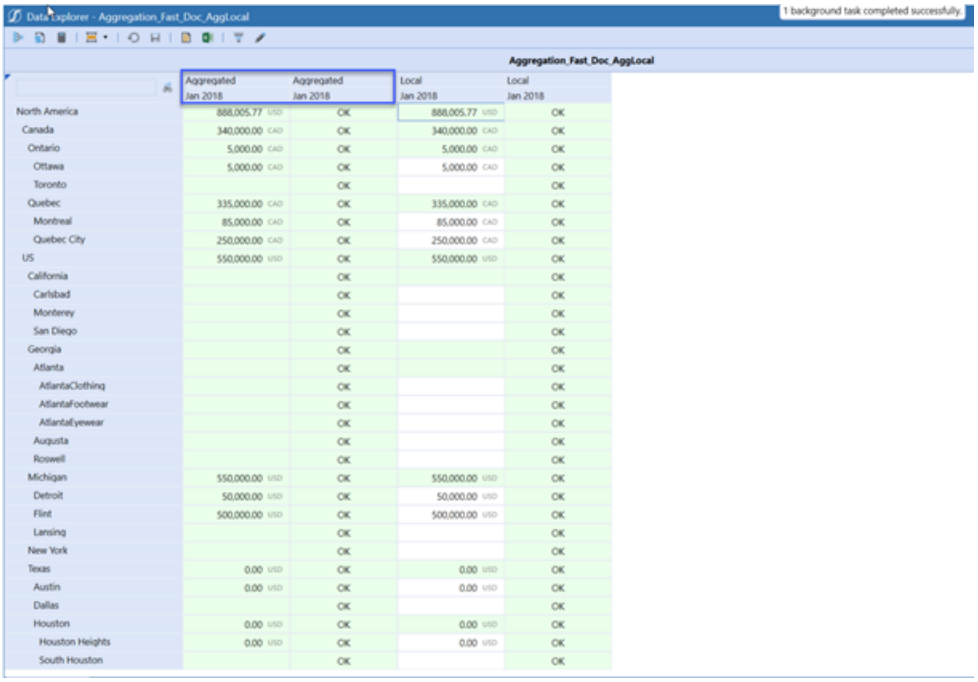


Example of Cube View Using C#Aggregated Dimension



- Rows
    - **Entity**: E#[North America].TreeDescendantsInclusive
- Columns

- **Consolidation**: C#Aggregated, C#Aggregated:V#CalcStatus, C#Local, C#Local:V#CalcStatus
- **Time**: T#2018M1, T#2018M12



Aggregate Jan 2018, by right-clicking on the number and pick Consolidate to Aggregate numbers. Context aware algorithms are used to recognize the aggregated member.

Notes:

- No Parent Journal Adjustments
- No Eliminations
- Business rules (execute at the base level only)
- Select different Consolidation member (C#Aggregated) to run consolidation on and pick the numbers that were loaded.

Launching an Aggregation

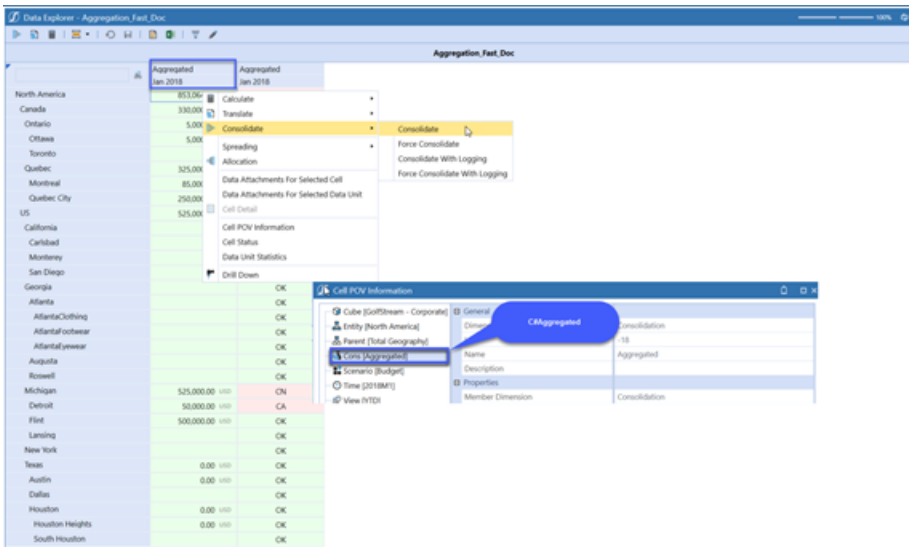
This is the same process as launching a consolidation using context aware algorithms once the aggregated member of the consolidation (C#Aggregated) is selected. For example, right-click on the appropriate cell in a Cube View or form to view the process by selectingConsolidate.



Consolidate

- Consolidate
- Force Consolidate
- Consolidate with Logging
- Force Consolidate with Logging

Aggregate Jan 2018, by right-clicking on the number and picking Consolidate to Aggregate numbers. Context aware algorithms are used to recognize the aggregated member.



Entity Aggregation Input

Entity aggregation input lets you submit data for base member values through the aggregated member. Use the Allow Input Into the Aggregated Consolidation Member property to enable input. Values entered into an aggregated cell are saved to the respective local currency member (C#USD, C#CAD, and so forth).

> **IMPORTANT:** Data entered through journals cannot be entered into the aggregated member.

This feature is available in Platform version 7.4 and higher.

Enable Entity Aggregation Input

1. On the Application tab, select **Dimensions**.
2. Expand **Scenario Dimensions**.
3. Expand the appropriate dimension, then select **Scenarios**.
4. On the Members tab, expand **Hierarchy** and select the aggregated membe.
5. Select the **Member Properties** tab.
6. Under Settings, set **Allow Input into the Aggregated Consolidation Member** to True.
7. Select **Save**.

Working With Hybrid Scenarios

Hybrid scenario data improves query performance if you analyze smaller data sets from large data units that contain high volumes of account-level detail for a cube such as scenario and entity. This enhances query performance when:

- Different business units require smaller subsets of the data unit because they only need to report on specific accounts, cost centers, products and so on.
- Business processes require a higher level view of data for budget or planning.

In these cases, data is filtered to focus on business and consumption needs for more efficient data analysis.

Hybrid data uses data from a source scenario member and displays the results in a target scenario member. You may need multiple target scenarios based on how you use hybrid source data in your application.

For information about associated properties, see Hybrid Scenarios. Also see:

- Data Bindings
- Data Filters
- Share Data From the Source Scenario
- Copy Data From the Source Scenario

Data Bindings

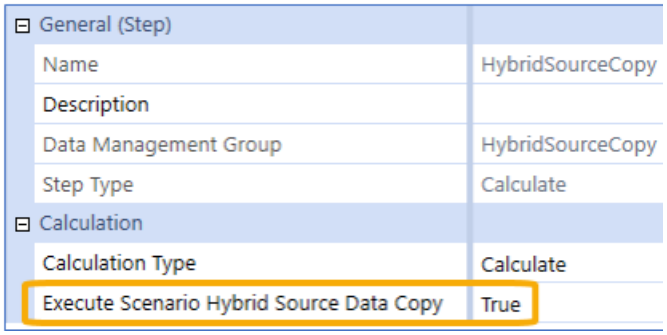
Data bindings determine if data is shared or copied from the source scenario member to the target scenario member. Sharing data is best if you need to analyze smaller data sets from a large data unit.

The data results are dynamic and reference source data. This data is not stored to the target scenario and is read-only. Shared scenarios also share the source scenario's calculation status and indicate source data changes. Standard calculations run from a shared scenario also run on the source scenario member.

Use copy data to compare what if scenarios, for budget versioning or forecasting. By default, a data copy occurs if a standard calculation - associated with the following items is run on the target scenario:

- Cube View or Quick View grid
- Calculation definitions
- Calculate data management step

The Calculate data management step offers more control over the hybrid source data process. Set **ExecuteCopyAfterCalc Scenario** to **False** and enable the copy execution on the data management step. If enabled, calculations run on the scenario, but the data copy only occurs if the calculation runs from the data management step.



Calculations follow the standard sequence and store the data as calculated. If multiple calculations run, the previously calculated or copied data is cleared. To preserve existing data, set the data as durable on the target scenario.

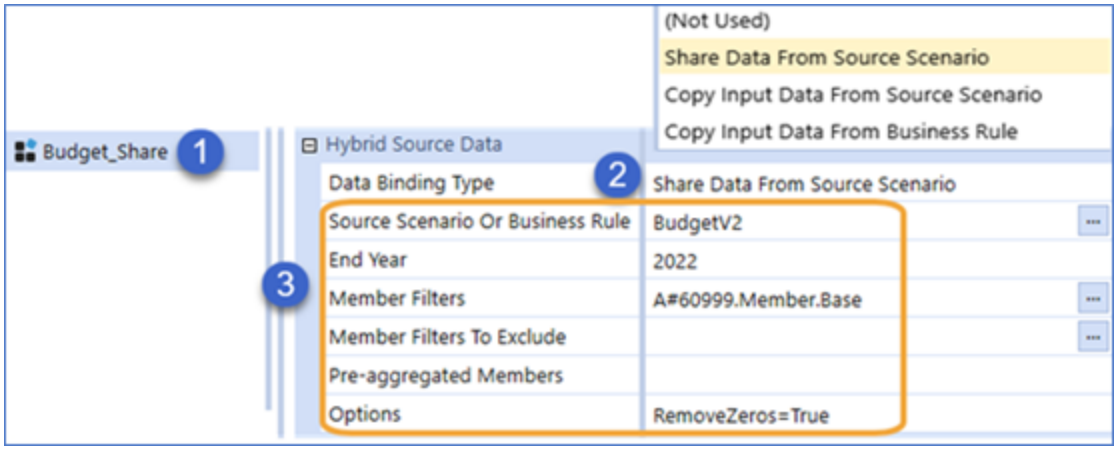
Once the target scenario copy finishes, you can modify and adjust data, but the target scenario is still bound to the source. The data copy occurs whenever the scenario is calculated based on the scenario setup. Hybrid copies can be chained, so results in a target scenario can be used as the source in another.

Data Filters

Hybrid source data generates unique data views using different filters, so you can best focus your data modeling and analysis.

Pre-aggregated members provide a summarized source data view by defining the source parent members and target base members. Members are filtered before aggregation and cached in RAM. Then the smaller data set is brought in and share or copy the information over to the target accounts.

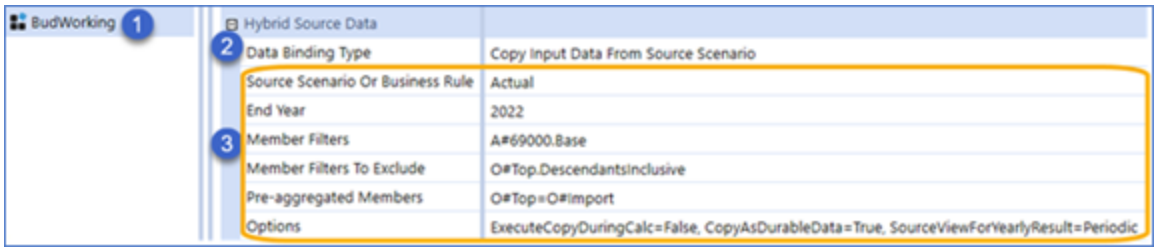
Share Data From the Source Scenario



1. Create a target scenario called Budget_Share.
2. Specify a Data Binding Type: Share Data from Source.
3. Select source scenario and data settings:
    - Source scenario: BudgetV2
    - Data binding ends in 2022

This shares the base accounts for Account 60999. Any zeros in the source data are removed.

Copy Data From the Source Scenario



1. Create a target scenario called BudWorking.
2. Specify a Data Binding Type: Copy Input Data from Source Scenario.
3. Select the source scenario and data settings:
    - Source scenario: Actual
    - Data binding ends in 2022

This copies the base accounts of account 69000 and excludes origin dimension members. Data is copied from the top of Origin in the source scenario to the Import member of Origin in the target scenario. The calculation occurs only if run from the data management step, and the copied data is stored as durable.

Time Profiles

Time profiles are used to align an organization's financial accounting period with Calendar or Fiscal Years for reporting purposes.

Time profiles are created under **Application** > **Cube** > **Time Profiles**. Many time profiles can be created to support an organization's financial accounting reporting needs. By default, one time profile is provided, called Standard. The standard time profile is configured as a calendar month to follow the January – December logic of the Gregorian calendar. The standard time profile cannot be deleted, and it is recommended not to change its configuration. In cases where part of an organization does not follow the standard calendar year for reporting, a new time profile is required for the fiscal year reporting needs.

> **Example:** Company A is a manufacturing company that conducts their financial reporting on a calendar year basis from January – December. Company A is looking to expand their footprint in the retail industry, and they purchase retailer Company B. Due to the nature of Company B's sales activities, Company B capitalizes on the heavy sales cycle and increased revenues during the holiday season. Therefore, Company B has a February – January fiscal year accounting period. In this example, a new time profile would be created to address Company B's February – January Fiscal Year accounting period for their reporting needs.

Anatomy of a Time Profile

The time profile consists of multiple sections:

- Time Profile Toolbar
- Profile Tab
- Time Periods Tab

Time Profile Toolbar

The time profile toolbar contains icons to support time profile functions.

1. **Create Time Dimension Profile**  : Use this iconto create a new time profile. When a new time profile is needed, selecton this icon to start the configuration process for a new time profile.

2. **Delete Selected Time Dimension Profile**  : Use this iconto delete a selected time profile. The intention of this iconis to delete a time profile that is unused or for time profiles that have not been assigned to a cube. Once the time profile is assigned to a cube and/or data has been loaded into the cube, the time profile cannot be deleted. This iconwill not be activated if the standard time profile is selected. Since the standard time profile is created by default, it cannot be deleted.

3. **Rename Selected Time Dimension Profile**  : Use this iconto rename a selected time profile. If the time profile is assigned to a cube and the time profile is renamed, the name change will be reflected in the cube. This icon will not be activated if the standard time profile is selected. Since the standard time profile is created by default, it cannot be renamed.

4. **Cancel All Changes Since Last Save**  : Use this iconwhen in the process of making changes to an already configured time profile and realize that the changes should not apply to the time profile.

5. **Save**  : Use this iconafter changing any properties in an existing time profile.

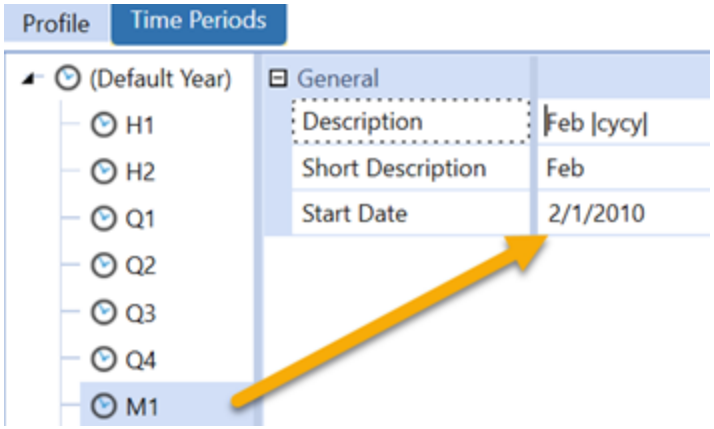
Profile Tab

- **Name**: The name of the time profile follows a free-form naming convention and is limited to 100 characters. Generally, there is no best practice for the naming convention. However, the name should be meaningful enough to identify the fiscal year, and it is worth considering whether the organization's name should be included in the new time profile. Consider whether a single time profile will be used across multiple cubes. This might necessitate a more generic naming convention that suits the needs of multiple cubes.
- **Description**: Include a short description for the time profile. The description is free-form and is limited to 200 characters.
- **Fiscal Year Start Date**: The start date of this time profile’s fiscal year.
- **Fiscal Year is First Period’s Calendar Year**: This property defines whether the fiscal year is based on the first period of the calendar year. The default setting for this property is False.

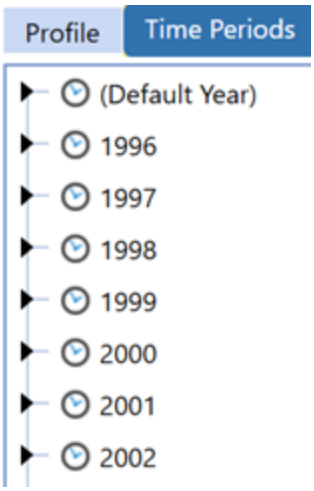
- **Fiscal Year Month Type**: This property defines how to align the month type with the fiscal year. The options are calender Month, Calendar Quarter, Fixed Weeks 4-4-5, Fixed Weeks 4-5-4, Fixed Weeks 5-4-4, and Custom Start Dates:
    - **Calendar Month**: Follows the standard 12-month Gregorian calendar
    - **Calendar Quarter**: Used in cases where data collection and reporting are performed on the quarters. As with Calendar Month, Calendar Quarter follows the standard 12-month Gregorian Calendar
    - **Fixed Weeks 4-4-5**: A weekly calendar determined by four weeks in first month, four weeks in second month, and five weeks in third month within a Quarter
    - **Fixed Weeks 4-5-4**: A weekly calendar determined by four weeks in first month, five weeks in second month, and four weeks in third month within a Quarter
    - **Fixed Weeks 5-4-4**: A weekly calendar determined by five weeks in first month, four weeks in second month, and four weeks in third month within a Quarter

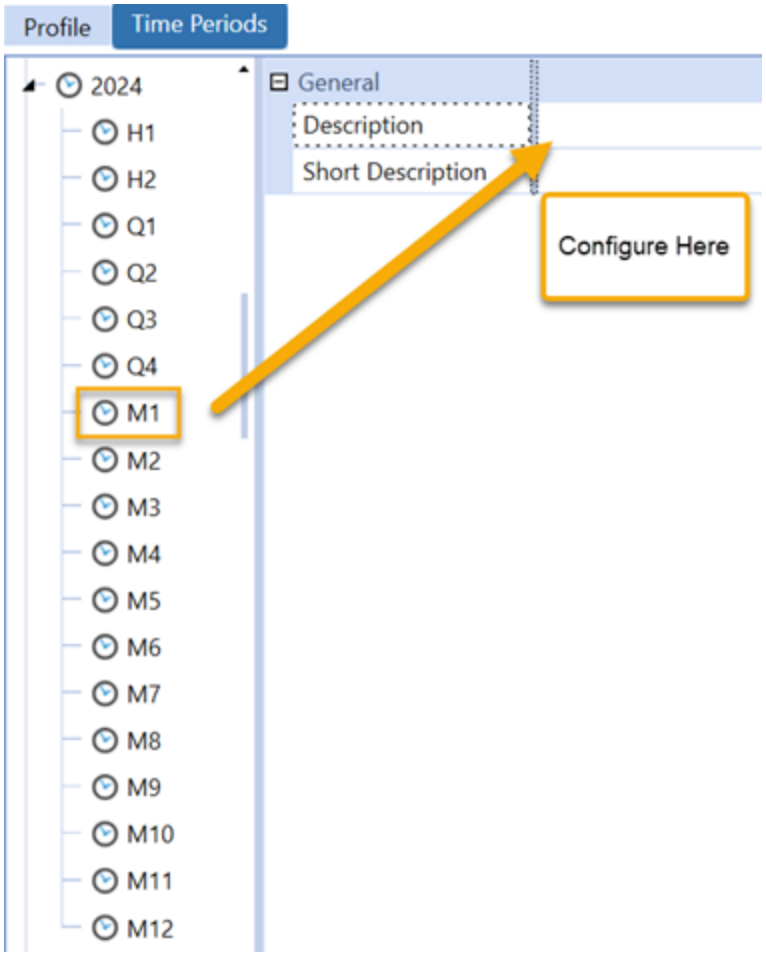
> **NOTE:** For Fixed Weeks, each month is either 28 or 35 days except for the twelfth month which will have an extra one or two days.

- **Custom Start Dates**: This option enables customization of the start date for each month. When this option is selected, the start date field is activated for each month in the Time Periods tab.



- **Vary Settings by Year**: This option allows time descriptions to be varied by year. When this option is set to True, each year is added to the Time Periods tab with time dimension members listed as children.





Each time dimension member under a year can have a unique description defined. By default, the time members that exist under the Year parent will be blank and ready to be configured if necessary. Otherwise, the Time members will assume the configuration settings set at the half year, quarter, month, week levels under the (Default Year) parent.

Time Periods Tab

The Time Periods tab is used to align the time periods, which are defined by the Time dimension configured at application creation time, with the time profile. Each time profile can be unique based on different fiscal year accounting periods. However, the time periods are defined at the application creation level. Therefore, different fiscal year time profiles would share the same time periods configuration. See Time Dimension.

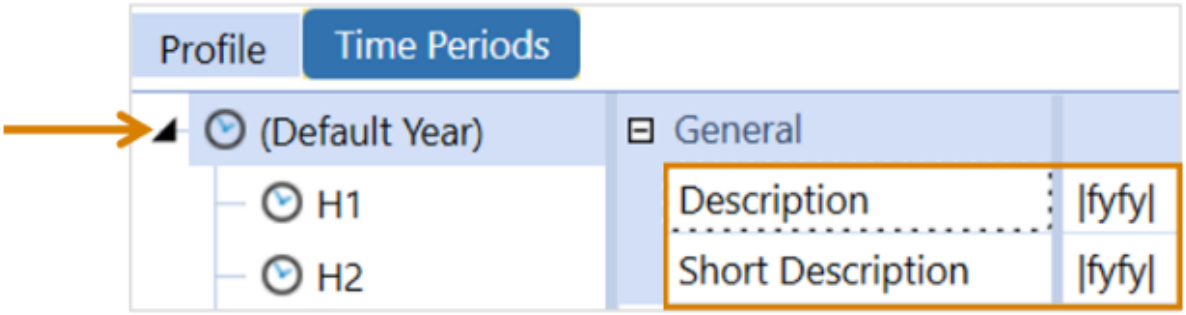
When configuring a time profile, a common question is how the time periods should be defined for the description and short description fields in the Time Periods tab. By default, the description field contains either |fyfy| or |cycy|. The half and quarters use |fyfy|, while months use |cycy|. If the application uses weeks, then weeks will use |fyfy|. Here, |fy| is a variable for fiscal year and |cy| is a variable for calendar year. If you use the variable format |fyfy| or |cycy|, the year will be displayed as a four-digit number, such as 2023, 2024, 2025. If you use the variable format |fy| or |cy|, the year will be displayed as a two-digit number, such as 23, 24, 25. The variable formats work in tandem with the Fiscal Year Start Date and the Fiscal Year Is First Period's Calendar Year properties on the Profile tab.

Default Year

Year Description: Displays the year as either fiscal year or calendar and defaults as |fyfy|.

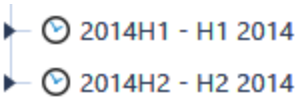


Short Description: Displays the year as either fiscal year or calendar and defaults as |fyfy|.

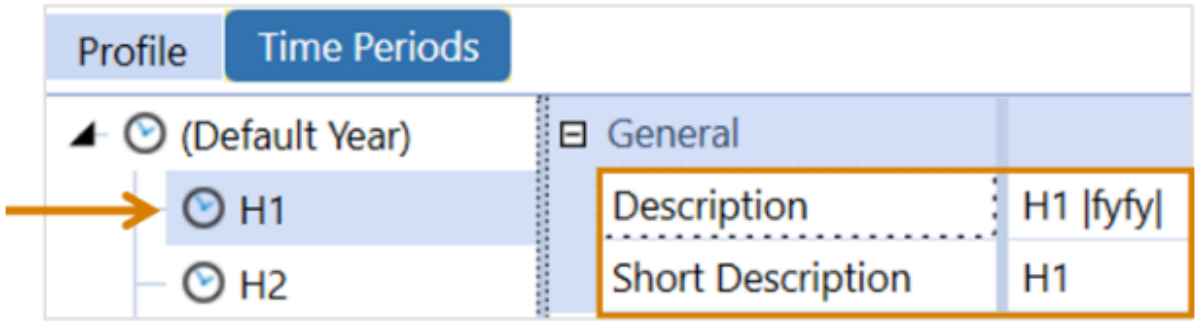


Half Years

HY1, 2 Description: Displays the half year as either fiscal year or calendar year and defaults as H1 |fyfy|.

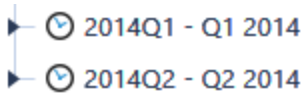


HY1, 2 Short Description: Displays the half year's short description and defaults as H1.

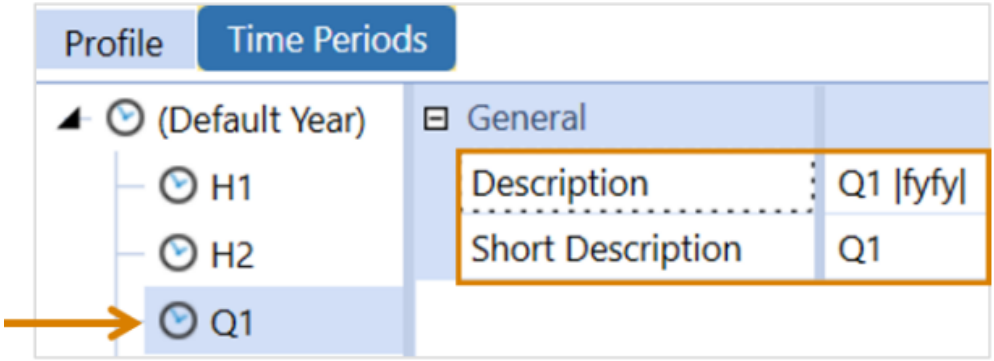


Quarters

Q1, 2, 3, 4 Description: Displays the quarters as either fiscal year or calendar year and defaults as Q1 |fyfy|.



Q1, 2, 3, 4 Short Description: Displays the quarters short description and defaults as Q1, Q2, Q3, Q4.

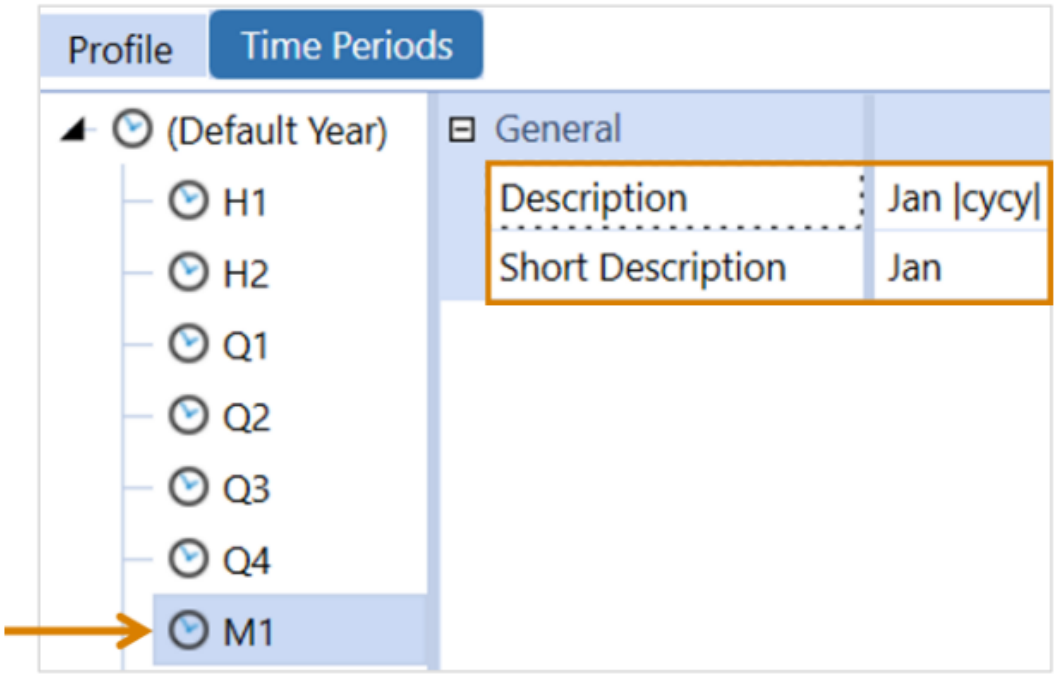


Months

M 1-12 Description: Displays the months as either fiscal year or calendar year and defaults as this format Jan|cycy|.



M1-12 Short Description: Displays the months short description and defaults as Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.



Weeks

Weeks are available if the application is created to accommodate weeks. Weeks are configured differently to half, quarter, and months. Not only do the weeks have description and short Description, but weeks also contain number of days for configuration. If the application contains weeks, here is how the weeks will be displayed.

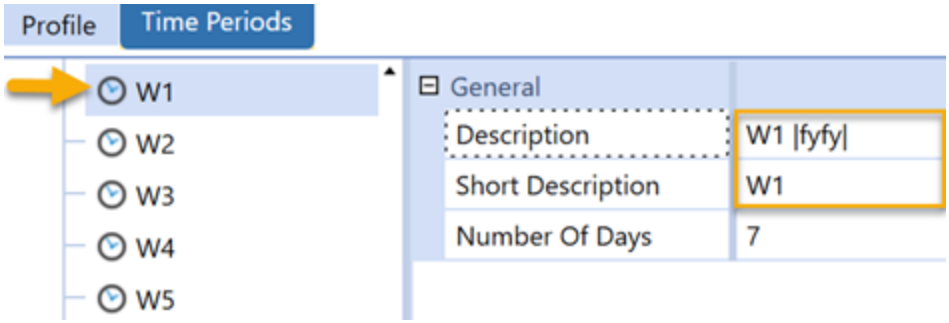
See Time Dimension, and find the Application Section for more information.

Description: Displays the weeks as either fiscal year or calendar year. Defaults to the |fyfy|.

2024W1 - W1 2024

2024W2 - W2 2024

W1-52 Short Description: Displays the weeks short description and defaults as W1 format.



Number of Days: Identifies how many days are considered for the week.

Cubes

Cubes are organization structures that contain data. They control how data is stored, calculated, translated, and consolidated based on dimensions assigned to the cube. While flexible and designed to hold multiple types of data, they are generally designed for specific purposes. An application can have several cubes that share dimensions, time profiles, business rule functions, and data. In this section, you will learn about cube dimension data, time profiles, and other cube-specific characteristics.

Cubes

Cubes are multidimensional organizational structures that contain data for reporting and analysis. A cube structure is formed through metadata defined as cube dimensions. It consists of 18 dimensions that are shaped and structured through Extensible Dimensionality. This allows it to store, calculate, translate, and consolidate data.

It is flexible, designed for specific purposes, and data types in a OneStream application. They are used to store data for reporting purposes and to leverage metadata structures through dimensions within a BI Blend process.

Applications can have multiple cubes that assist in overall performance and reporting efficiencies. Each can be unique or share similar characteristics. Cubes are created under **Application** > **Cube** > **Cubes** and are set up and controlled within the cube profile.

Structure of a Cube

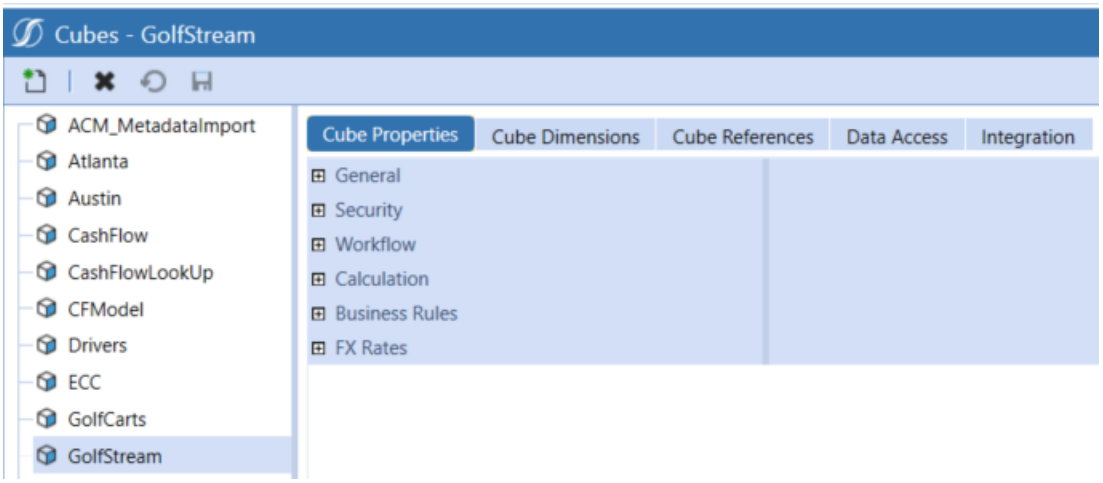
A cube consists of multiple tabs. Each tab plays a key role supporting the structure and processing:

- Cube Properties
- Cube Dimensions
- Cube Reference
- Data Access Tab
- Integration

See Dimensions Overview.

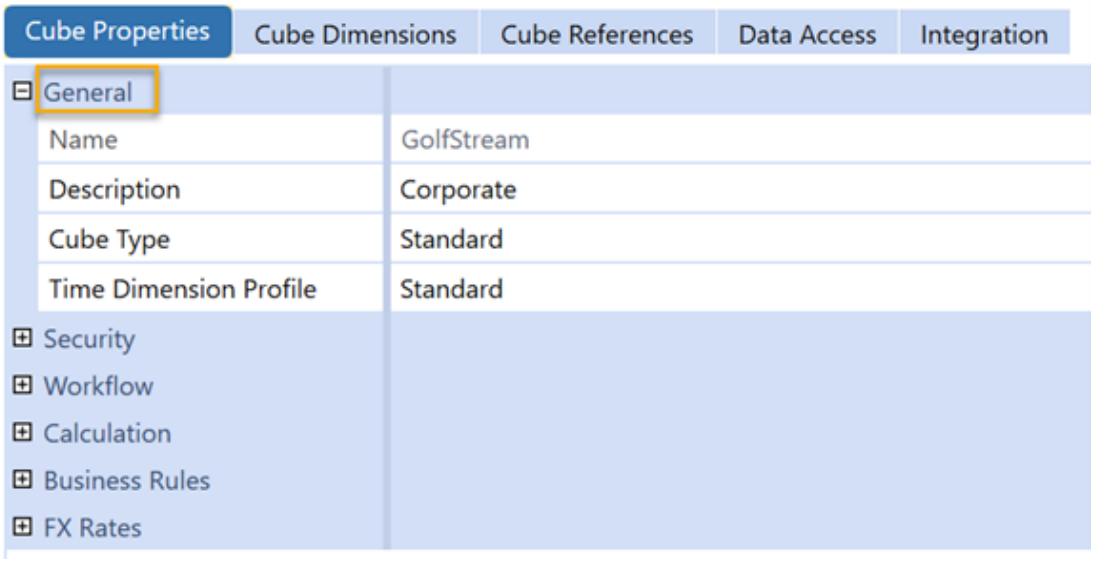
Cube Properties

You can view and edit cube properties in **Application** > **Cube** > **Cubes**. The Cube Properties tab enables you to assign initial key settings that impact how a cube interacts with a workflow, and is is secured, calculated, translated.



General

Inside the Cube Properties tab is a section called General.



Name: field to name a cube and limited to 100 characters

> **NOTE:** When determining a naming convention for a cube, it cannot be changed once it has been created. Consideration should be used when using a company name as the cube name since company names can change either intentionally or through merger.

The cube Name property can be accessed through the finance engine using a Finance business rule:

- Dim objCubeInfo As CubeInfo = api.Cubes.GetCubeInfo(name)
- Dim objCube As Cube = objCubeInfo.Cube
- Dim sValue As String = objCube.Name

Description: field to provide a description for the cube and limited to 200 characters. The cube description can be changed at any time after creation.

The cube Description property can be accessed through the finance engine using a Finance business rule:

- Dim objCubeInfo As CubeInfo = api.Cubes.GetCubeInfo(name)
- Dim objCube As Cube = objCubeInfo.Cube
- Dim sValue As String = objCube.Description

Cube Type:This optional setting creates tags to group cubes with similar characteristics. This is used to have different settings for Default and Constraint settings that apply to specific dimensions and vary by cube type, such as entity. The cube type names are random and do not have functional differences but represent various kinds of cubes that may be created. Below are all Cube Types property options:

- **Standard**: Default option for cube type when a new cube is created.
- **Tax**: This option is considered for a cube that is focused on a tax process. A new cube can be created for a tax process while using dimensions assigned to a cube with a standard cube type.
- **Treasury**: This option is considered for a cube that is focused on a treasury process. A new cube can be created for a treasury process while using dimensions assigned to a cube with a standard cube type.

> **NOTE:** When using the same dimensions across cubes with different cube types, the constraints can be changed to allow different dimension members from existing dimension to be the constraint member by cube type. This applies for Tax or Treasury properties.

- **Supplemental:** This option is considered for additional data for the cube type. Supplemental data can be part of a process captured during an Actual, Budget, or Planning process. Other factors can determine whether a new cube is needed, and it is not necessarily driven by supplemental data, but it may be included anyway. In many cases, supplemental data is not specifically assigned to a cube with a supplemental cube type and is not always a strict rule. Supplemental is an option used to group cubes together that use the same dimension member constraints. The characteristics of the supplemental data and how it should be constrained dictate whether supplemental is a viable option to use for cube type on the cube.
- **What If:** This option can be considered when forecasting different hypothetical models for a business. Cubes can be created using existing dimensionality. However, for a What If Cube, the constraints, or defaults may differ than a cube processing an Actual scenario.
- **Cube Type 1…Cube Type 8:** This option can be considered as grouping to align cubes together with common characteristics compared to other cubes using similar dimensionality.

The cube type property can be accessed through the finance engine using a Finance business rule:

- Dim objCubeInfo As CubeInfo = api.Cubes.GetCubeInfo(name)
- Dim objCube As Cube = objCubeInfo.Cube
- Dim nValue As Integer = objCube.CubeTypeId

The cube type is stored in a column within the dbo.Cube table as an integer (Int). Each cube type is assigned a unique integer value:

- Standard = 0
- Tax = 1
- Treasury = 2
- Supplemental = 3

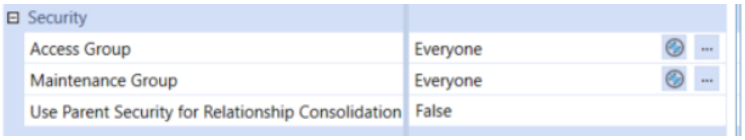
- What If = 4
- Cube Type 1 = 11
- Cube Type 2 = 12
- Cube Type 3 = 13
- Cube Type 4 = 14
- Cube Type 5 = 15
- Cube Type 6 = 16
- Cube Type 7 = 17
- Cube Type 8 = 18

Time Dimension Profile: This property provides a drop-down menuof available time profiles. The time profile assigned to the cube aligns accounting calendars to the time periods for reporting and processing for the cube.

See Time Profiles.

Security

This section controls who maintains and has access to the cube. It also controls data access to consolidation members at relationship levels.



Access Group: The security group can view the object and its related contents. The default setting is Everyone. The order of security to gain access to data starts from gaining access to an application then to a cube. As there are more layers of security to gain access to data, the default setting is Everyone and additional security is assigned at lower layers. You will determine if a security group should be assigned to the Access Group to control access.

Maintenance Group: The security group can view the object, create new objects in groups, edit, and delete.

Use Parent Security for Relationship Consolidation: This property controls the security model for the read and write relationship for the consolidation dimension members. This works with the security rights to the immediate parent-level entity. This property defaults to False. The default behavior is that anyone who has access to parent-level entities will also get access to all the consolidation dimension members. When this property setting is changed to True, you can have access to Local and Translated consolidation dimension members but no longer have access to parent-level relationship consolidation dimension members such as Share, Elimination, OwnerPostAdj, and Top. This is determined on a cube-by-cube basis.

Workflow

This section controls the availability and management of workflow cube root structures for Workflow Profiles for a cube.

Workflow	
Is Top Level Cube For Workflow	True
Suffix for varying Workflow by Scenario Type	
Actual	ACT
Administration	
Budget	PLAN
Control	
Flash	
Forecast	PLAN
FXModel	
History	
LongTerm	
Model	
Operational	
Plan	PLAN

Is Top Level Cube for Workflow: When a new cube is created, this property setting defaults to False. When set to False, the cube is unable to create and maintain a workflow structure. When set to True, the cube can create and maintain a workflow structure. This is determined on a cube-by-cube basis.

During metadata requirements and design, the number of cubes needed are evaluated. If using Extensible Dimensionality with multiple linked cubes, 1 cube will have this setting set to True, while the remaining cubes will have this setting set to False. Only 1 cube will be responsible for controlling and maintaining Workflow Profiles and structures through a cube root profile for Workflow Profiles.

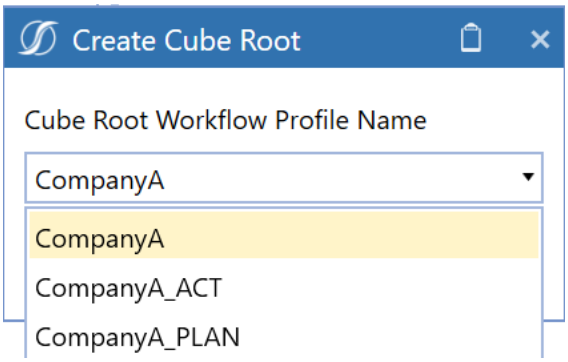
Suffix for Varying Workflow by Scenario Type: This property is used to group scenarios by scenario types to make available to users in the workflow. This can be determined on a cube-by-cube basis. This is active only for cubes with Top Level Cube for Workflow set to True. There is one setting for each of the scenario types. This property can act as a layer of security by organizing and grouping scenarios with certain scenario types to the appropriate user groups who are responsible for certain processes in an organization.

By default, each scenario type is blank when a cube is created. If the Is Top Level Cube for Workflow property setting is set to True, then a Workflow Profile cube root can be created for a cube. When the Workflow Profile cube root is created, any Workflow Profile that is created will have all the scenario types available to load data to by default. This means the same Workflow Profiles and Workflow Profile hierarchies are used for all the scenario types.

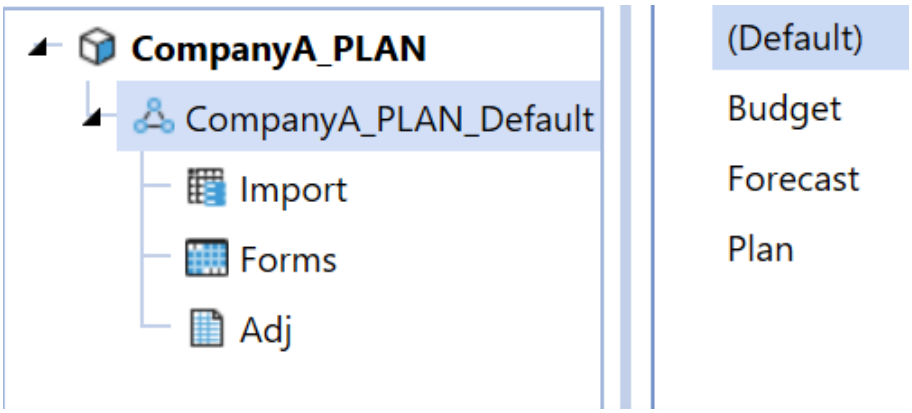
While technically feasible, this design may not offer optimal efficiency for organizing Workflow Profiles and Workflow Profile hierarchy structures. For example, different users are responsible for different processes within an organization. Accounting/Finance is responsible for Financial Close & Consolidation, while Financial Planning and Analysis (FP&A) is responsible for Budget, Planning, and Forecast. To support this example of organizational responsibility, Suffix for Varying Workflow by Scenario Type configuration may display like this:

By configuring the Suffix for Varying Workflow by Scenario Type for budget, forecast, and plan with PLAN, the scenario types are now grouped together to share the same cube root Workflow Profile and Workflow Profile hierarchy structure. Any scenario assigned a scenario type of budget, forecast, or plan can load data through the PLAN cube root Workflow Profiles.

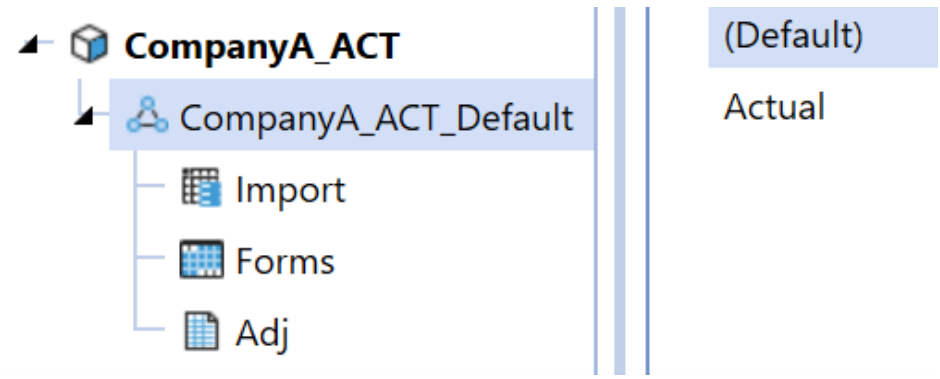
Based on the configuration on the cube, the cube root Workflow Profile names would show accordingly when creating a cube root:



After creating the cube root for CompanyA_PLAN, the scenario types available for the PLAN Workflow Profiles would look like this:



After creating the cube root for CompanyA_ACT, the scenario types available for the ACT Workflow Profiles would look like this:



At the time of designing and configuring the cube, it is not necessary to know what to name the suffix for each scenario type.

> **IMPORTANT:** Once a scenario is defined with a scenario type and the scenario contains data, a cube root Workflow Profile cannot be created for that scenario type and the property setting field for the scenario type will not be considered for a new cube root. See "Workflow" in the *Workflow Guide*.

> **Example:** For example, in the first phase of a project, which may be Financial Close and Consolidation, it is determined that a scenario called ACTUAL will be assigned the actual scenario type. The actual scenario type field in Suffix for Varying Workflow by Scenario Type is set to ACT, but all the other scenario types are left blank. Once the workflow design and build are completed, data is loaded into the ACTUAL scenario.

During the project, a requirement may necessitate the seeding of actual data into a Budget scenario assigned the Budget scenario type. Once data is seeded into the Budget scenario in the cube, the Budget scenario type can no longer have a suffix or be renamed in the Suffix for Varying Workflow by Scenario Type. A new cube root will not be created for the Budget scenario type if the budget process becomes more developed in a later phase.

Data and all processes related to the Budget scenario must be deleted to make this change. A Reset Scenario Data Management step would need to run to reset the Budget scenario, and then a suffix can be added to the Budget scenario type. See "Data Management Export Sequences Data Source" in the *Data Collection Guide* for more information.

If data has not been loaded to a scenario/scenario type combination and the scenario type is blank, then a suffix name can be added to the scenario type and a new cube root can be created. This is independent of any other scenario/scenario type combination that may have data already loaded.

Calculation

This section guides how to process, calculate, and translate data under certain conditions.

Calculation	
Consolidation Algorithm Type	Standard (Calc-On-The-Fly Share and Hierarchy Elimination)
Translation Algorithm Type	Standard
Calculate None Cons Member If No Data	False
Calculate Local Currency If No Data	True
Calculate Translated Currencies If No Data	False
Calculate OwnerPreAdj If No Data	False
Calculate Share Cons Member If No Data	False
Calculate Elimination Cons Member If No Data	False
Calculate OwnerPostAdj If No Data	False

Consolidation Algorithm Type

When a consolidation runs in OneStream, data is moved up to the Entity and Consolidation dimensions. The consolidation algorithm type on the cube specifies how the share and elimination members will be treated.

Standard (Calc-on-the-fly Share and Hierarchy Elimination): Standard (Calc-On-The-Fly Share and Hierarchy Elimination) is the most used consolidation algorithm type. This cube setting calculates an entity's share amount dynamically. The value does not get stored in the application database data record tables. Share is a consolidation dimension member defined for a specific parent/child relationship and is calculated as an entity's (translated balances + owner pre-adj journals * percent consolidation). Eliminations under standard are calculated usingOneStream's built-in algorithms.

Stored Share (Stored Share and Hierarchy Elimination): The Stored Share consolidation algorithm type stores the amounts for share rather than calculating them dynamically as in standard. However, the eliminations under stored share are still calculated as in standard, using OneStream's built-in algorithms. This consolidation algorithm type can be used when you need different logic in calculating share than in standard. For example, if you had a minority interest calculation, where the share contribution cannot be driven from the percent consolidation.

When the Stored Share consolidation algorithm type is used, the rules to calculate the value need to be written under the Finance Function Type ConsolidateShare within a Finance business rule.

Org-By-Period Elimination (Calc-on-the-fly Share and Org-By-Period Elimination): The Org-By-Period Elimination algorithm type uses the calc-on-the-fly share – as in standard – but has unique elimination considerations. When determining if the data cell's IC member is a descendant of the entity being consolidated, this consolidation algorithm type considers the position of the entity in the hierarchy and checks the percent consolidation for every relationship down the hierarchy. If percent consolidation is zero for the relationship, the IC member is determined not to be a descendant of the entity.

In comparison, the standard elimination (hierarchy elimination) only considers the position of the member in the entity dimension hierarchy. Standard elimination is the default approach and does not consider percent consolidation.

Stored Share And Org-By-Period Elimination: The Stored Share and Org-By-Period Elimination Consolidation algorithm type is a combination of the two settings which have been explained above.

Custom: The Custom Consolidation algorithm type allows for the ability to calculate share and elimination data intersections using logic within a finance business rule. This is often used when you have custom eliminations that are different fromOneStream's built-in algorithm and org-by-period elimination logic. This could be due to having custom eliminations within a user-defined dimension or wanting to write eliminations to unique and specific data intersections.

There are two finance function types within a finance business rule in which the logic is written: FinanceFunctionType.ConsolidateShare and FinanceFunctionType.ConsolidateElimination.

Performance Considerations Using Consolidation Algorithm Types: To determine what consolidation type settings are needed for the application, it is also important to be aware of performance implications. Under the standard consolidation algorithm type, OneStream does not store share because it is dynamically calculated. When turning on any of the above consolidation algorithm types which store share, there will be performance implications.

In storing share, the consolidation engine must perform the custom logic written in the Finance business rule and write the data records to the data record tables in the database. The size of each data unit will become larger because of the additional data records stored in the data record tables in the database.

Elimination data records are stored in the data record tables in the database under all consolidation algorithm types. When turning on custom eliminations, performance implications will depend on the rule design and the construction of the entity hierarchy. Additional data records may be written to the data record tables in the database and the engine may spend more time in the custom elimination logic than OneStream built-in algorithms.

Translation Algorithm Types

When a consolidation runs in OneStream, data is moved up to the Entity and Consolidation dimensions. The translation algorithm type on the cube specifies how data records during the translated consolidation members will be treated, such as:

Standard: Standard is the most used translation algorithm type. This cube setting takes an entity's local currency values and translates them based on the FX rate types such as, average, closing, and more. Also, FX rule types, such as periodic, direct, and more, are assigned to the scenario.

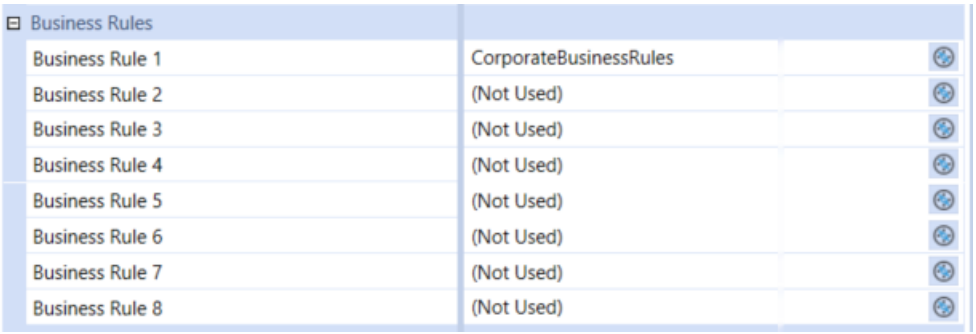
Standard Using Business Rules for FX Rates: Standard Using Business Rules for FX Rates is like standard but allows the ability to use a business rule to specify translation rates for any given intersection. Any intersections not specified in the Finance business rule will translate based on the standard translation logic. This is commonly used during the translation of forecast, constant currency, and other such scenarios.

It is also used when the rate needed for translation already exists in the FX rate table but in another rate type/time, or the rate needs to be determined dynamically. For example, consider needing to translate the actual scenario at the current year's budget rates. In this case, all the actual data needs to be translated based on the rates which already exist in the FX rate table. By using a business rule, youcan dynamically determine which year's rates the translation is based on without having to copy rates that have already been entered to another rate type. This reduces administrator maintenance by eliminating the need to copy or enter duplicate rates within the system.

Custom: Custom translation logic is rarely used but allows for the ability to calculate the translated values for all intersections within a Finance business rule. The system is flexible to modify to custom unique methods that are not common or out of the box.

Business Rules

Business logic can be introduced at the cube level. This can be accomplished by attaching a Finance business rule to the cube in the business rules section. The same Finance business rule logic can be shared among cubes. Eight business rules can be defined and processed in a logical order with Member Formulas. These run on Calculate, Translate, or Consolidate processes, seeConsolidation.

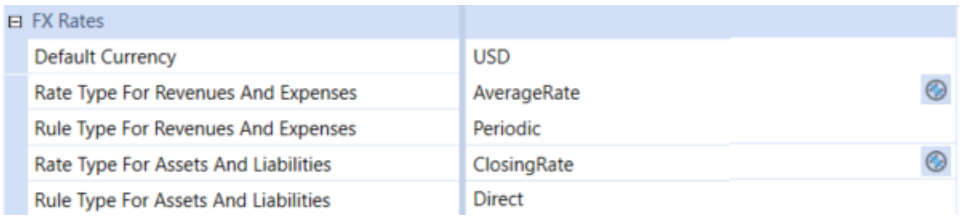


Business Rules	
Business Rule 1	CorporateBusinessRules
Business Rule 2	(Not Used)
Business Rule 3	(Not Used)
Business Rule 4	(Not Used)
Business Rule 5	(Not Used)
Business Rule 6	(Not Used)
Business Rule 7	(Not Used)
Business Rule 8	(Not Used)

BusinessRule1…BusinessRule8: This property uses a drop-down option for attaching a Finance business rule. The business rule must be created as a Finance business rule for the business rule to appear as a selection option.

FX Rates

This section provides the cube logic on how to process FX rates for specific account types and define the cube with a default currency to leverage for reporting and during the translate process.



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

Default Currency: This property is the default reporting currency for the cube. This is used for FX rate triangulation if the cube currency is the common currency and intercompany matching reporting currency.

Rate Type for Revenues and Expenses & Rate Type for Assets and Liabilities: The property settings can be configured at the cube level or overridden at the scenario level. These properties have pre-defined selection options out of the box. The pre-defined selection options for these properties are Average Rate, Opening Rate, Closing Rate, and Historical Rate. Although a pre-defined list of options is available, new FX rate types can be created in **Cube** > **FX Rates** which will instantly make the new FX rate type available for selection. See Foreign Exchange Rates for more information.

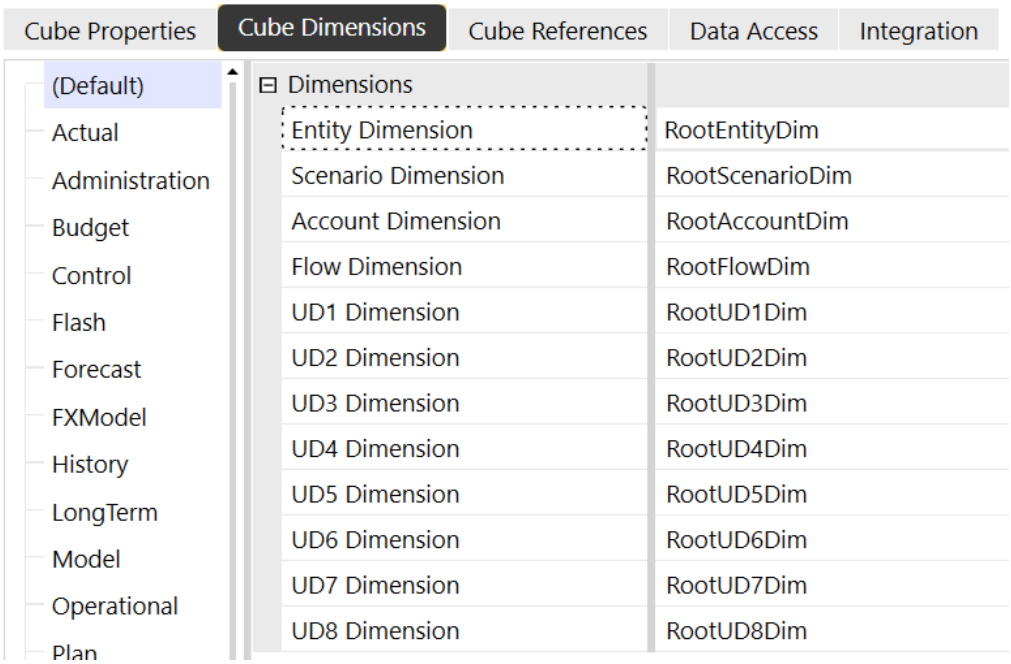
Rule Type for Revenues and Expenses & Rule Type for Assets and Liabilities: The settings for the properties can be configured at the cube level or overridden at the scenario level. These properties have pre-defined selection options out of the box. The pre-defined selection options for these properties are Direct and Periodic. As a default when a cube is created, the rule type for revenues and expenses is set to periodic while the rule type for assets and Liabilities is set to direct.

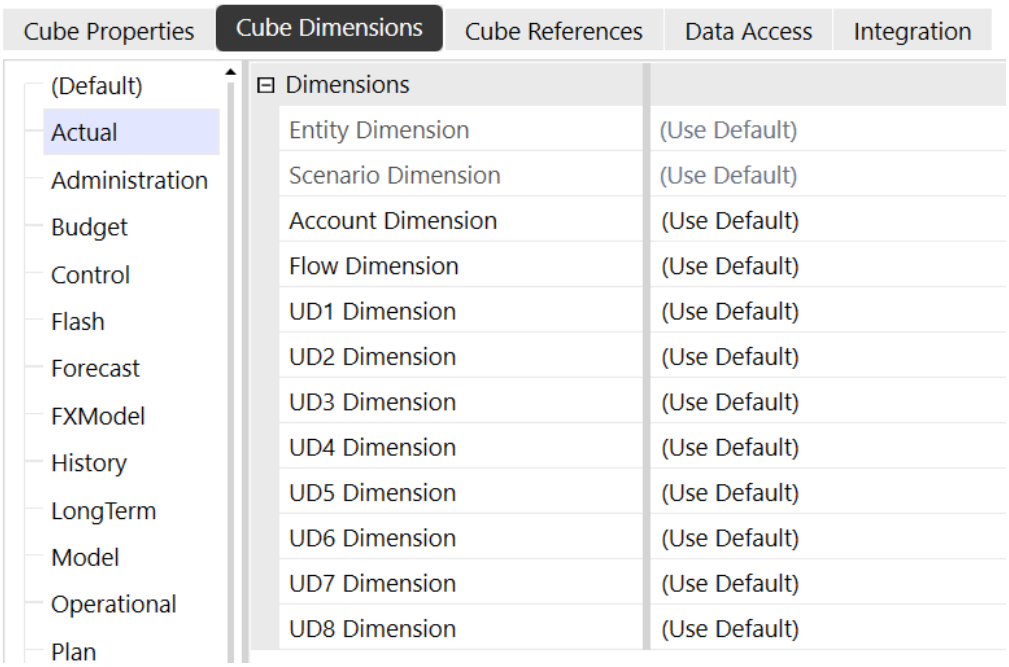
- **Direct**: Calculate direct with current value and current rate. Current Period's Translated Value = Current Period's Local Value * Current Period's FX Rate.
- **Periodic**: Calculate periodic value translation method. This method considers the translation rates for prior time periods and calculates the form of average. Current Period's Translated Value = Prior Period's Translated Value + [(Current Period's Local Value – Prior Period's Local Value) * (Current Period's FX Rate)].

Cube Dimensions

Dimensions are assigned to a cube in the Cube Dimensions tab. The dimension metadata provides structure to the cube and dictates how the cube will organize the data within the application. When assigning dimensions to a cube, it is best practice to assign the cube dimensions by scenario type, with the exception of Entity and Scenario dimensions, which can only be assigned in the (Default) scenario type.

The entity and scenario dimension types cannot be assigned at the scenario type level because they do not adhere to the same concept of Extensible Dimensionality as the account related dimension types. When creating a new cube, the (Default) option is set to RootXXXDim for all dimension types by default, where XXX represents the name of any dimension type. At the scenario type level, the entity and scenario types are grayed out by default and not configurable. All other dimension types are set to (Use Default).

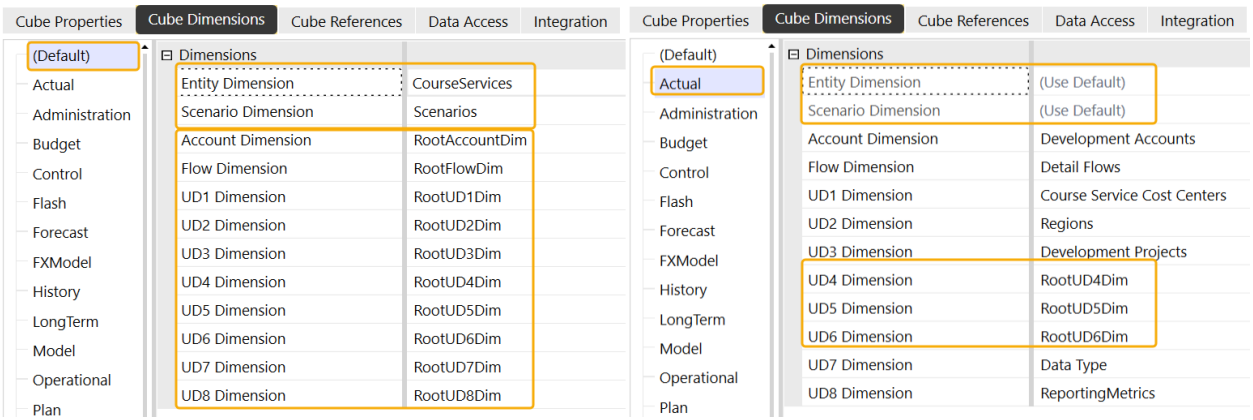




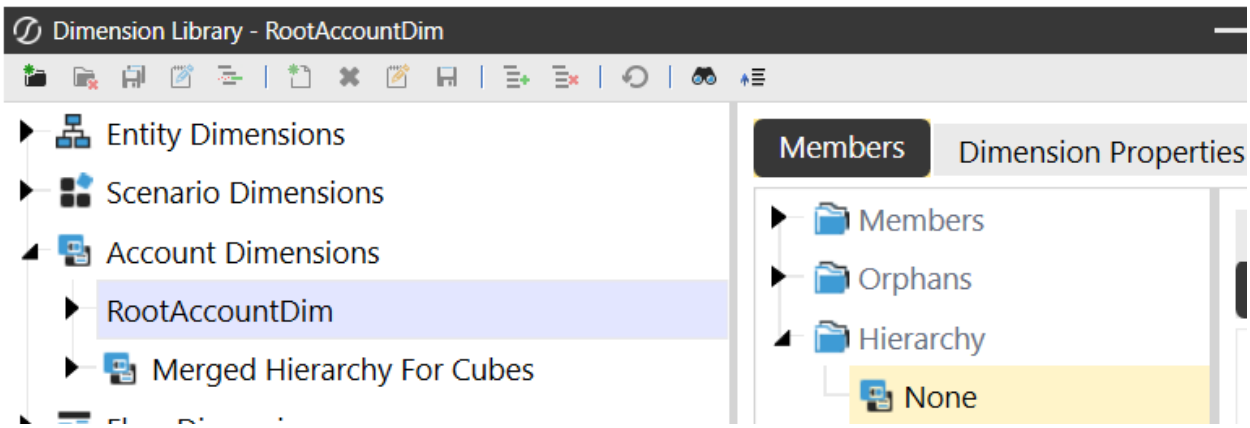
As a cube is being configured and dimensions are assigned, not all dimension types may be used for a specific scenario type. In cases where a scenario type does not use a certain dimension type, the RootXXXDim should be assigned to that dimension type instead of left at (Use Default).

> **NOTE:** While this adds flexibility for the future, configuring the cube dimensions this way does not enable you to change active dimensions with data.

The following examples display the recommended cube dimension configuration for(Default) and the actual scenario type. At (Default), the entity and scenario dimension types are assigned. All other dimension types are left at RootXXXDim. At the actual scenario type, the entity and scenario dimension types are grayed out and cannot be configured. UD4, UD5, and UD6 dimension types are not used in this example. Therefore, they are set to RootXXXDim.



Each dimension type contains a RootXXXDim, and inside RootXXXDim is a member called None. When a new dimension for a dimension type is created and RootXXXDim is selected as the inherited dimension, the dimension becomes a child of RootXXXDim and the new dimension inherits RootXXXDim and the None member.



Assigning RootXXXDim for dimension types inactive at setup instead of leaving (Use Default) is a recommended configuration concept, as it allows flexibility if the dimension type needs to be used in the future. If RootXXXDim is assigned to the dimension type and data is loaded to the cube, all data for that dimension will be in the dimension's None member, which is inherited from RootXXXDim. If a new dimension is created for the scenario type in the future, the dimension can be changed from RootXXXDim to the new dimension name for the dimension type. However, if the unused dimension type is set to (Use Default) for the scenario type and data has been loaded to the cube for the scenario type, the dimension is locked in and cannot be changed to a new dimension in the future. The (Default) scenario type can be changed from RootXXXDim, but this change would apply to the entire cube, not to a specific scenario type. See About the Financial Model

> **IMPORTANT:** You can only update from the Root dimension to a specific dimension once. Changing from a root dimension is a one-time change that cannot be reverted if there is data in the cube and scenario type combination. Plan and ensure that these settings are properly updated before saving the changes.

Once dimensions are assigned to the cube for a dimension type and data loads to dimension members in a cube, dimensions are locked in and can no longer be unassigned or changed. To change a used dimension type with a different dimension, the data must be cleared and all processes related to the cube and workflow for that scenario must reset. This can be done through a reset scenario data management step. See "Data Management" in the *Application Tools Guide* for more information.

Use Case

This Use Case demonstrates the following situation:

- **Scenario:** The user wants to add a customer dimension in the budget scenario type to expand the annual planning capabilities and include their top customers. Data has been loaded to Actual and a prior year Budget. This use case provides example configurations to display the recommended approach and common misconfigurations.
- **Role:** Administrator, Developer
- **Benefits:** Setting all unused dimension types to the root dimension enables you to add additional dimensions to specific scenario types in the future. If this is not configured properly, the entire cube must abide by the updates to the (Default) scenario type.

See Cube Dimension Assignment Use Case.

Cube Dimension Assignment Use Case

The following use case provides a more detailed look into cube dimension assignment.

Scenario: The user wants to add a customer dimension in the budget scenario type to expand the annual planning capabilities and include their top customers. Data has been loaded to Actual and a prior year budget. This use case provides example configurations to display the recommended approach and common misconfigurations.

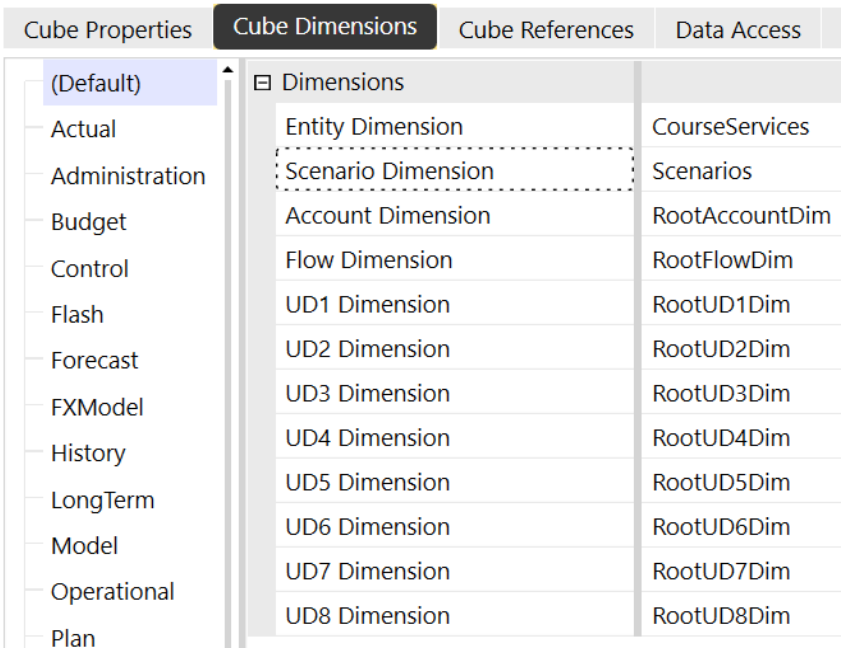
Role: Administrator, Developer

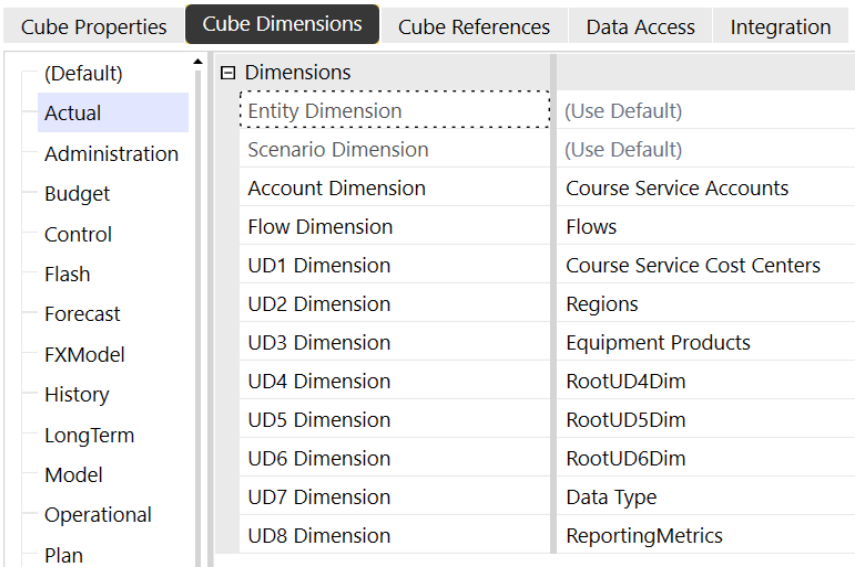
Benefits: Setting all unused dimension types to the root dimension enables you to add dimensions to specific scenario types in the future. If this is not configured properly, the entire cube must abide by the updates to the (Default) scenario type.

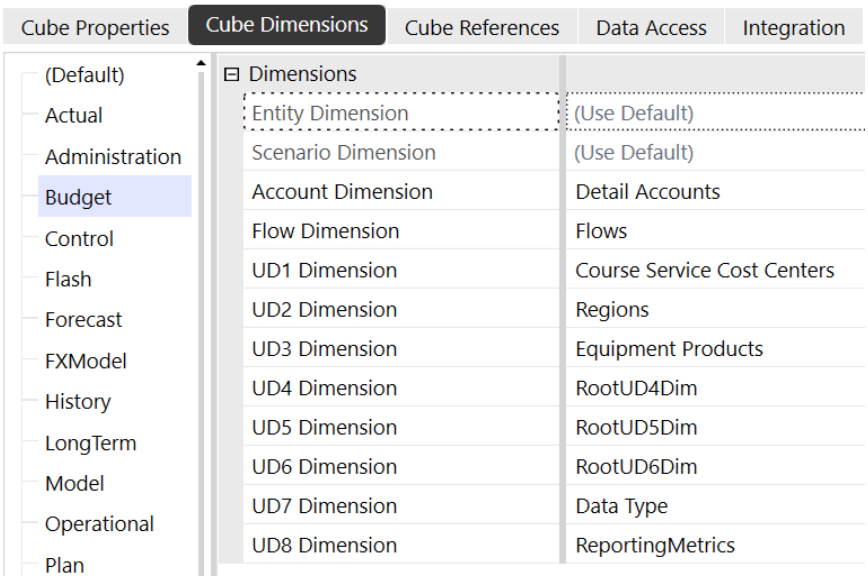
Recommended Configuration

When a cube is created, specific scenario types with dimension types set to (Use Default) should be updated to the root dimension. This enables data model flexibility and expansion in the future. For this use case, the scenario types are set to the following recommended configuration:

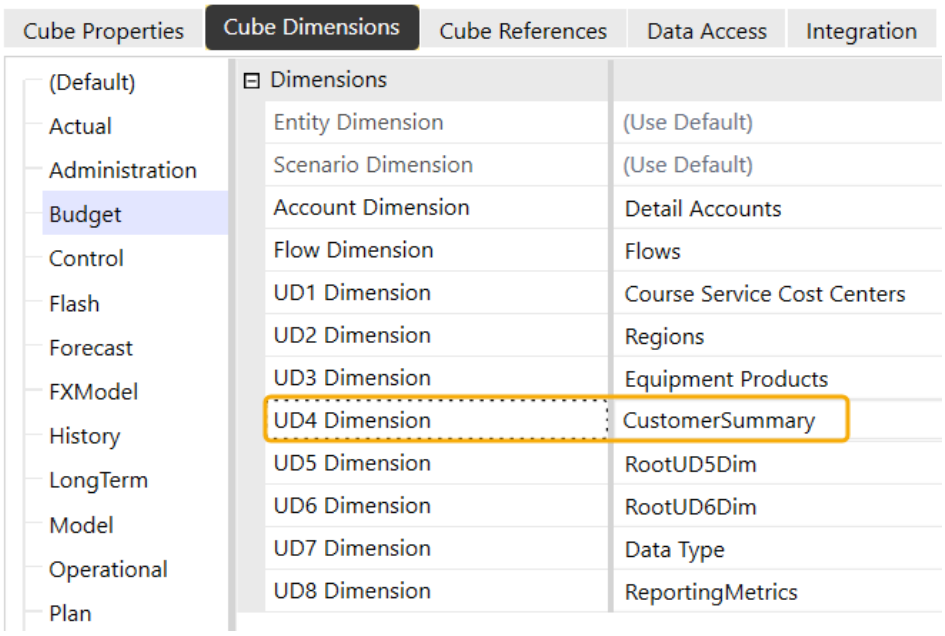
- In the (Default) scenario type, entity and scenario dimension types are assigned. All non-data unit dimension types are left at the root dimension.
- For all active scenario types, all non-data unit dimensions are assigned to a specific dimension. The root dimension is selected for all unused dimensions. (Use Default) is not set for any dimension type. Inactive scenario types remain unchanged until they are ready to be activated.







After the actual and budget scenario types have data in them, due to following the recommended configuration, the user is able to change the UD4 Dimension assignment in the budget scenario type to include the new summary customer dimension.



IMPORTANT: Once the new dimension type assignment is saved, the assignment is locked in. Changing from a root dimension is a one-time change that cannot be reverted if there is data in the cube and scenario type combination.

Once the new dimension is added to the budget scenario type, the user will see the history in UD4#None and that the new dimension members are active for input in subsequent budget cycles. Since it was assigned to the specific budget scenario type instead of the (Default) scenario type, the new UD4 dimension is invalid for the actual scenario type.



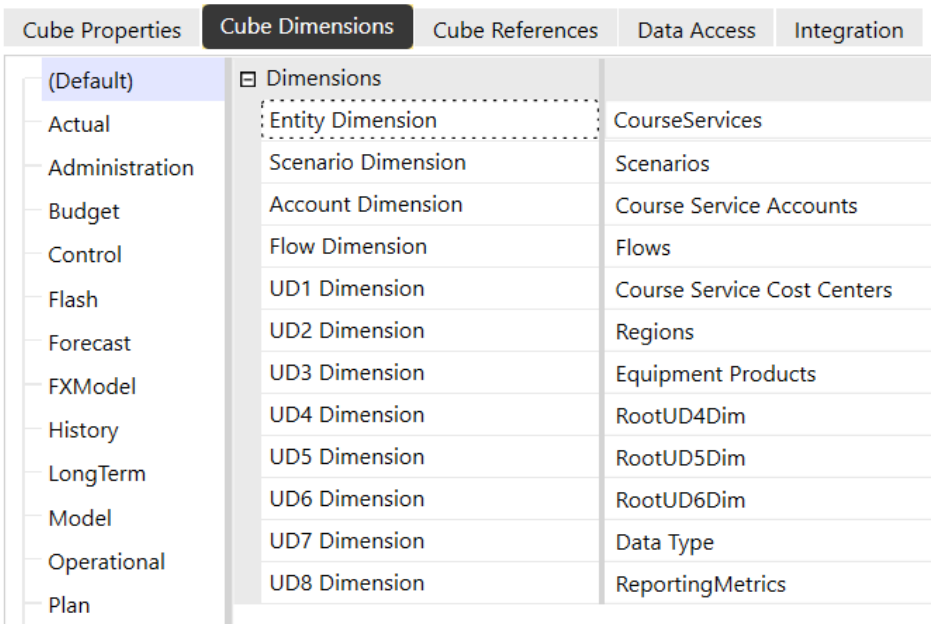
This configuration also enables the user to add UD4 and UD5 dimensions in the future by following the same steps.

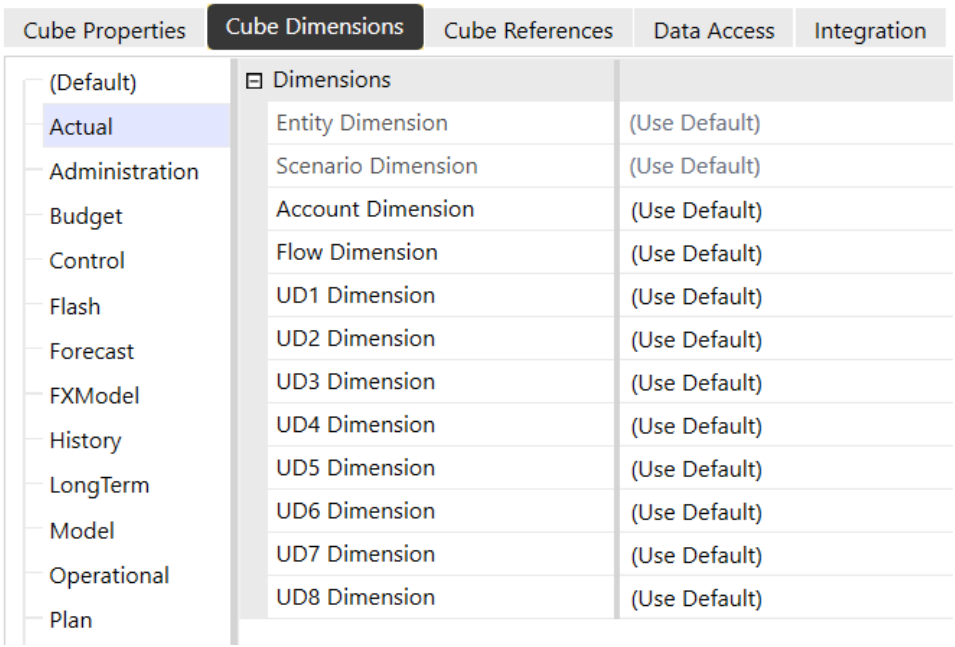
> **NOTE:** While this adds flexibility for the future, configuring the cube dimensions in this way does not enable you to change active dimensions with data.

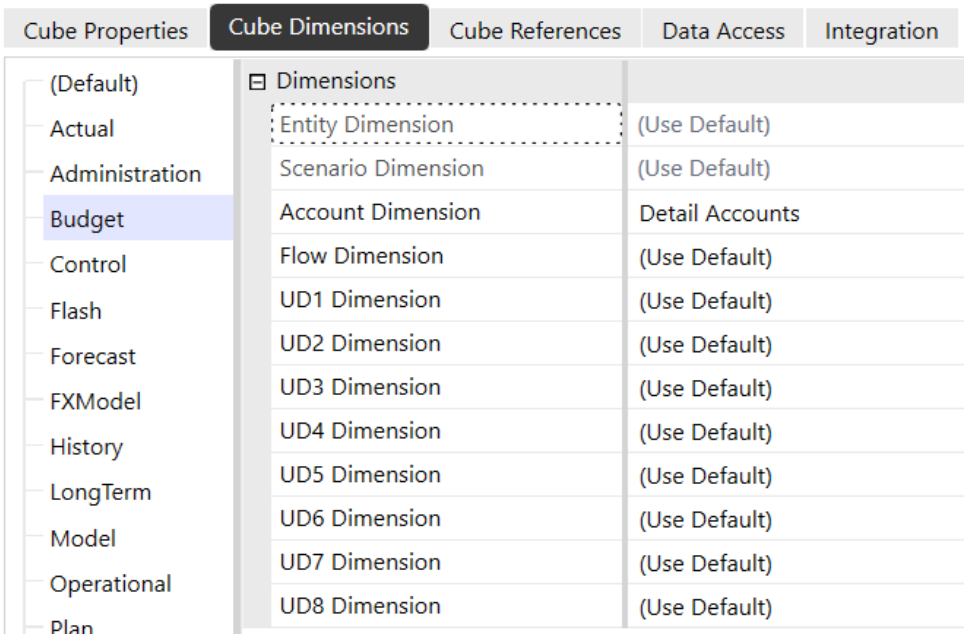
Common Misconfiguration

It is not recommended for the user to assign all dimensions to the (Default) scenario type and only use additional scenario types for dimensions that differ. While this configuration works and enables the user to add additional dimensions in the future, it is much less flexible. Additional dimensions in this setup must be assigned to the (Default) scenario type and will apply to all active scenarios. For this use case, the scenario types are set to the following non-recommended configuration:

- In the (Default) scenario type, entity and scenario dimension types are assigned. All active dimensions are also assigned to the (Default) scenario type.
- In the actual scenario type, all dimension types are left as (Use Default).
- In the budget scenario type, a different account dimension is added to enable the use of extensibility. All other dimension types are left as (Use Default).

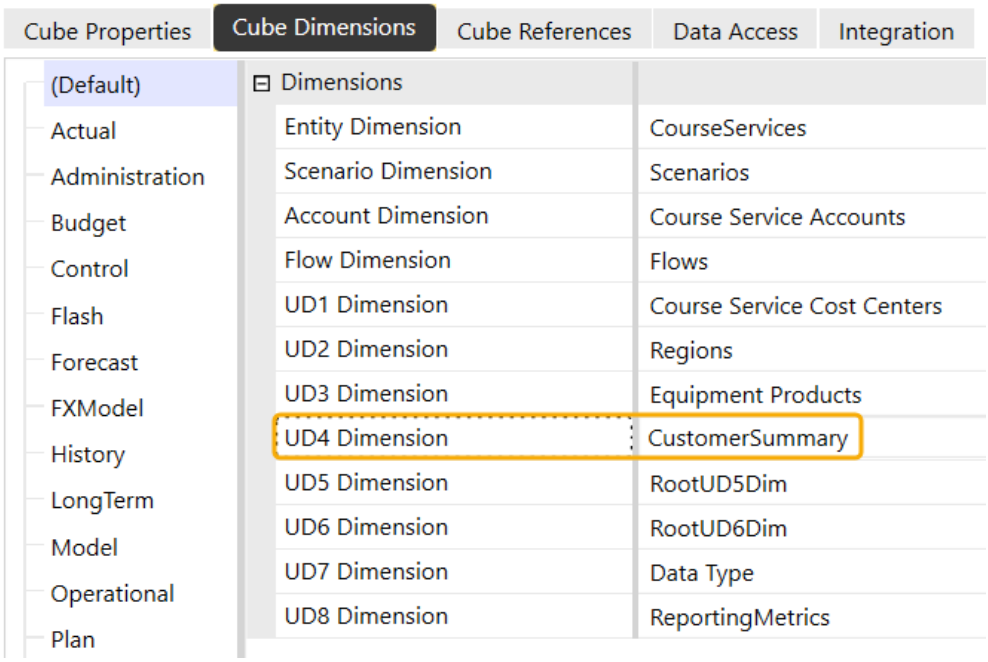

Cube Properties
Cube Dimensions
Cube References
Data Access
Integration
(Default)
Actual
Administration
Budget
Control
Flash
Forecast
FXModel
History
LongTerm
Model
Operational
Plan
Dimensions
Entity Dimension
CourseServices
Scenario Dimension
Scenarios
Account Dimension
Course Service Accounts
Flow Dimension
Flows
UD1 Dimension
Course Service Cost Centers
UD2 Dimension
Regions
UD3 Dimension
Equipment Products
UD4 Dimension
RootUD4Dim
UD5 Dimension
RootUD5Dim
UD6 Dimension
RootUD6Dim
UD7 Dimension
Data Type
UD8 Dimension
ReportingMetrics


Cube Properties
Cube Dimensions
Cube References
Data Access
Integration
(Default)
Actual
Administration
Budget
Control
Flash
Forecast
FXModel
History
LongTerm
Model
Operational
Plan
Dimensions
Entity Dimension
(Use Default)
Scenario Dimension
(Use Default)
Account Dimension
(Use Default)
Flow Dimension
(Use Default)
UD1 Dimension
(Use Default)
UD2 Dimension
(Use Default)
UD3 Dimension
(Use Default)
UD4 Dimension
(Use Default)
UD5 Dimension
(Use Default)
UD6 Dimension
(Use Default)
UD7 Dimension
(Use Default)
UD8 Dimension
(Use Default)

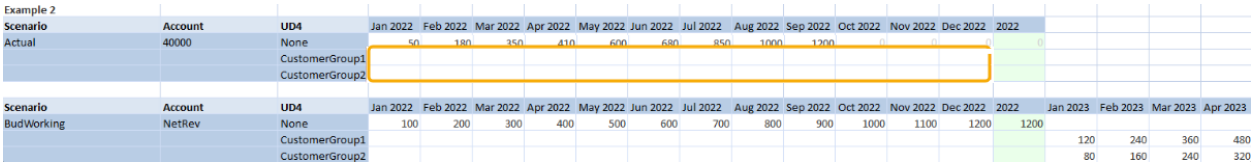


With this configuration, the user will receive an error if they assign the new customer dimension to UD4 in the budget scenario type. Because (Use Default) was left on all active scenario types, the user is now locked into whatever the (Default) scenario type has set for these dimensions, and they cannot be updated.

To add the new summary customer dimension, the user must assign it to the (Default) scenario type.



This assignment works for the budget scenario type as required, but it is now active for the actual scenario type, which may not be the desired result.

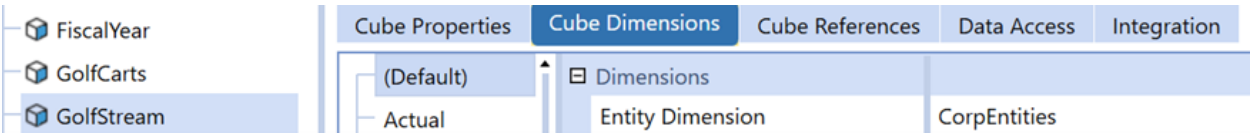


With this configuration, existing business rules and Member Formulas need to be validated throughout the application to ensure the right intersections are specified. Because this additional dimension contains valid intersections in all scenario types, rules need to be explicit in their filtering and writing of data. If rules are written improperly or too openly, this new dimension may impact performance and cause bad data to be calculated in these new intersections. Additionally, no input rules may be necessary to limit input to these intersections in scenario types that do not apply.

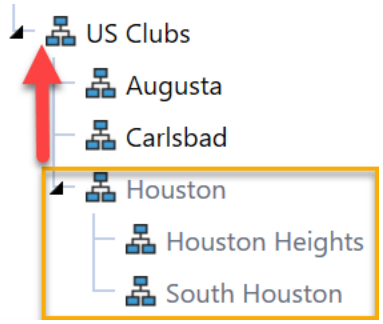
Cube Reference

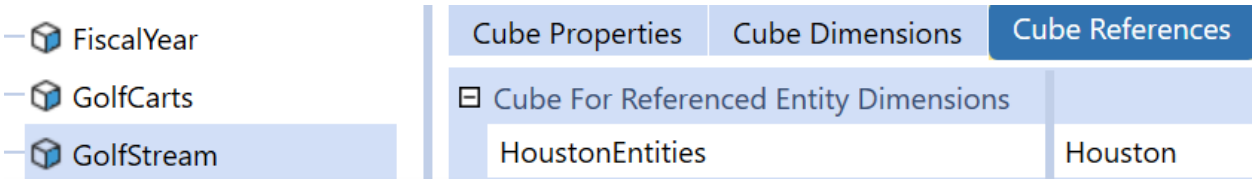
The Cube References tab is used to bring linked cubes together. By using cube references, one cube is defined as a parent cube. The parent cube will have the cube references configured to include all referenced Entity dimensions from child cubes. When a cube is created, the cube reference tab is blank. When Entity dimensions from child cubes have a relationship with the Entity dimension from a parent cube, the cube for referenced Entity dimensions section will populate for the parent cube. Creating these references links child cubes to the parent cube and ensures data consolidates into the referenced entity hierarchy of the parent cube. See Entity Dimension

> **Example:** In the example below, Golfstream is the parent cube and acts as a super cube. CorpEntities are the Entity dimension assigned to the Golfstream cube. CorpEntities are the Entity dimension that other Entity dimensions will create relationships with to create cube references.

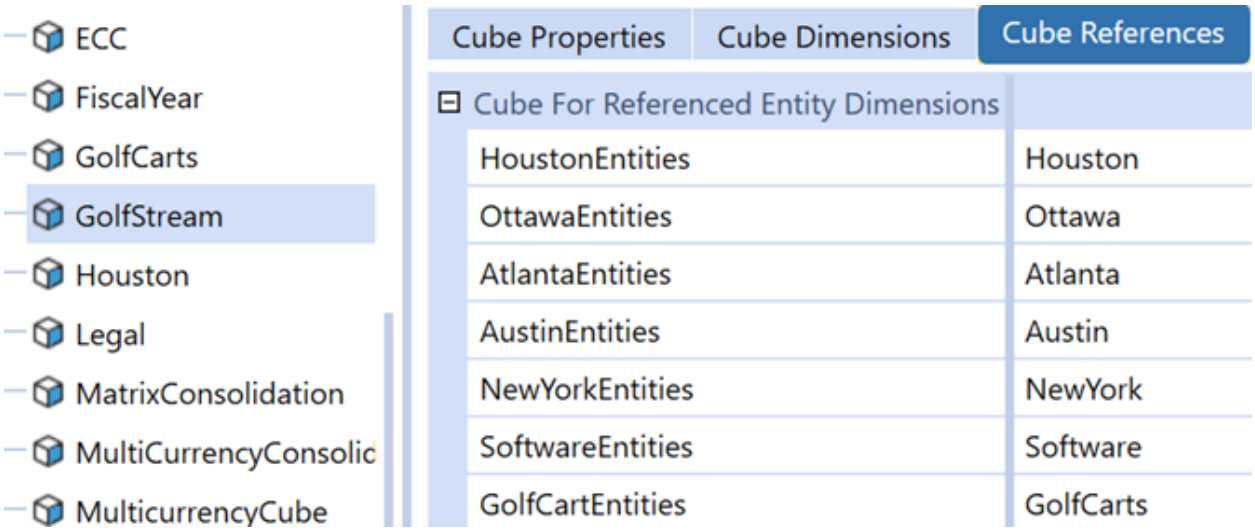


When an entity hierarchy from another Entity dimension, such as HoustonEntities, has a relationship with an entity in the CorpEntities Entity dimension, HoustonEntities will displayin the Cube Reference tab under Cube for Referenced Entity Dimensions for the Golfstream cube.





It continues to create relationships between Entity dimensions from child cubes to CorpEntities in Golfstream. A final list of referenced Entity dimensions in the Golfstream cube may display like this:



Data Access Tab

The Data Access tab controls access to data at the cube level. As part of the security order of operations, once a user has access to an application, the next layer of security is to get access to the cube. This is controlled in **Cube Properties** > **Security**. See Cube Properties.

Next layer of security is getting access to the data stored in a cube. When considering using data access, the security level gets down to slices of data/data buffer or even down to a more granular level of the data cell intersection.

The Data Access tab contains three sections:

- Data Cell Access Security
- Data Cell Conditional Input
- Data Management Access Security

Data Cell Access Security

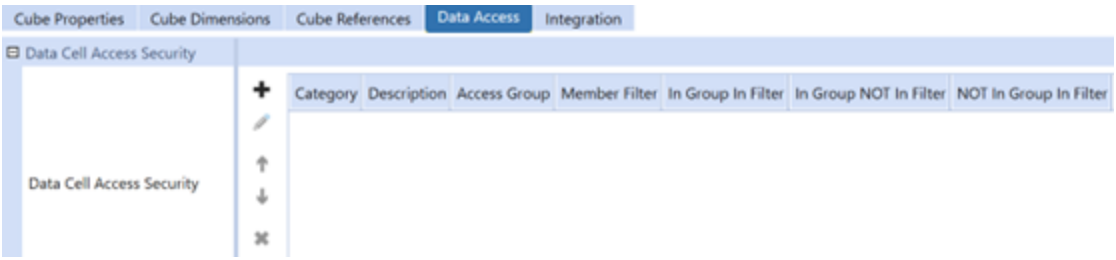
This section is also considered Slice Security. Here is where access rules can be created to decrease or increase access to data at a more granular level than **Application** > **Cube** > **Entity** > **Scenario**. No Access, Read Only, or All Access can be granted to a group of data cells down to a single data cell.

To get access to entity data, you must have read/read write access to the entity through entity security.

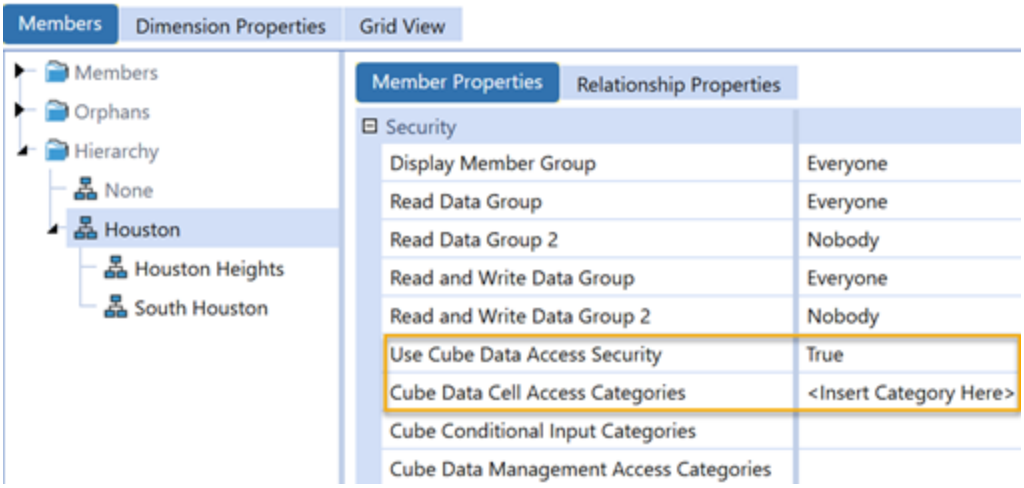
When granted access to the entity data, you get access to every single data cell for the entity within the cube. If there is a situation where the user should not have access to every data cell for the entity, then Data Cell Access Security is where configuration would happen to control access to these data cells.

Choose a user group, the level of access, and then enter a Member Filter. For example, a user group that includes senior management and human resources can have All Access to actual compensation figures (S#Actual, A#[Total Compensation].Tree), but everyone else will have No Access.

- **Category**: This is an optional category name by which access rules can be named and grouped. The naming convention for category is free form and is limited to 100 characters. When a cube is created, by default, the data cell access security section is blank.



If these categories are created, more than one can be applied to an entity's security settings. Enter an optional comma-separated list of category names that will be used when processing the cube's slice-level data access algorithms. Use empty text to process all categories. Include |Null| to represent nameless categories.



To add a category to **Cube Data Cell Access Categories**, the **Use Cube Data Access Security** must be changed to **True**.

- **Description**: This is an optional free form field to add a description for the data access rule. The description is limited to 200 characters.
- **Access Group:**This is a security group in which users are assigned to where data cell access security roles apply.

The access group shows a selection list of security groups from security framework. The access group can be either manually created security groups or system defined security groups such as EntityReadDataGroup, EntityReadWriteDataGroup, and so on.

Security groups are created through security and each security group is assigned a unique identifier.

- **Member Filter:**Member Filter is used to define the dimensions and dimension members required as part of the data cell access to secure or allow access.

The Member Filter could be a singular dimension and dimension member or a combination of multiple dimensions and dimension members. By using Member Filters, the data cell access security roles control access to a subset of data. This is considered Slice Security.

The Member Filter uses the same Member Filter syntax as used throughout the application.

- **Action**: The action section is broken out into three different actions:
    - If the user is in Group and Data Cell is in Filter.
    - If the user is in Group and Data Cell is not in Filter.
    - If the user is not in Group and Data Cell is in Filter.

For each of these three actions, the behavior and access level can be defined for a data cell access security role. Based on the action case, a series of behaviors and access levels will apply.

In a data cell access security role, the behavior has eight available behaviors to choose from. Each behavior has a unique role in how a user gains access to data or how a user is restricted from data. The eight behaviors are:

- **Skip Item and Continue:** Default for If User is in Group and data cell is not in filter or If User is not in Group and data cell is in filter.
- **Skip Item and Stop**: Choose this behavior to skip a cube data access Item and stop evaluating the remaining cube data access items.
- **Apply Access and Continue:** Default for If User is in Group and data cell is in filter
- **Apply Access and Stop:** Choose this behavior to apply access to a cube data access item and stop evaluating the remaining cube data access items.
- **Increase Access and Continue:** Choose this behavior to increase access to a cube data access item and then continue evaluating the remaining cube data access items.
- **Increase Access and Stop:** Choose this behavior to increase access to a cube data access item and then stop evaluating the remaining cube data access items.
- **Decrease Access and Continue:** Choose this behavior to decrease access to a cube data access item and then continue evaluating the remaining cube data access items.
- **Decrease Access and Stop:** Choose this behavior to decrease access to a cube data access item and then stop evaluating the remaining cube data access items.

In conjunction with the eight behaviors, there are three different access levels associated with them. The Member Filters, behaviors, and access level all work together to define data cell access.

- **No Access**: This access level prevents users from Read or Write access to cells defined in the Member Filter.
- **Read Only**: This access level allows users Read access to cells defined in the Member Filter.
- **All Access**: This access level allows users Read and Write access to cells defined in the Member Filter.

All these components make the data cell access security. Data cell access security is performed after access to **Application** > **Cube** > **Entity** > **Scenario**.

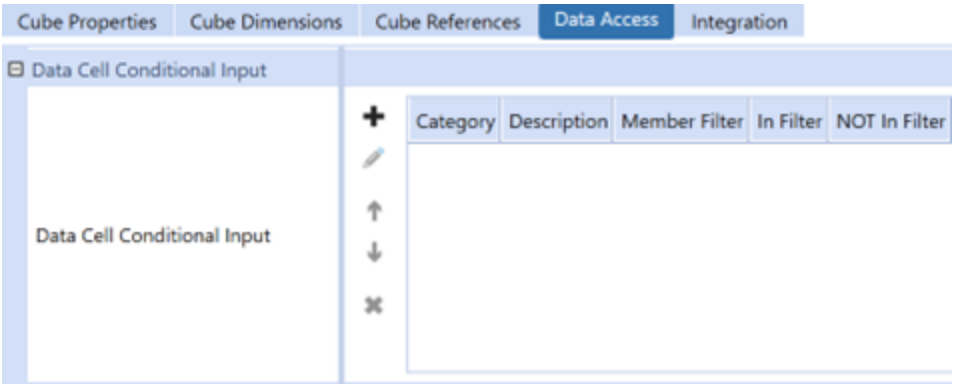
Data Cell Conditional Input Security

This section enables you to conditionally provide behavior access for all users to input data for a group of data cells or a specific data cell.

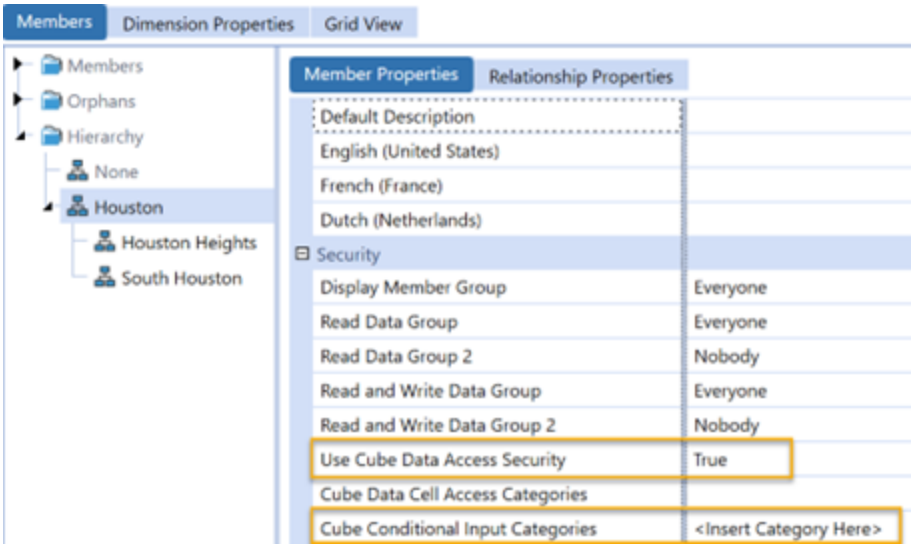
Data cell conditional input shares the same properties and behaviors as data cell access security. However, data cell conditional input does not have an access group property. When configuring the Member Filters for a data cell conditional input rule, all users will either have access or not have access to input data to the data cells defined in the Member Filter.

Structure of Data Cell Conditional Input Security

- **Category**: This is an optional category name by which access rules can be named and grouped. The naming convention for category is free form and is limited to 100 characters. When a cube is created, by default, the data cell access security section is blank.



If these categories are created, more than one can be applied to an entity's security settings. Enter an optional comma-separated list of category names that will be used when processing the cube's slice-level data access algorithms. Use empty text to process all categories. Include |Null| to represent nameless categories.



To add a category to **Cube Conditional Input Categories**, the **Use Cube Data Access Security** must be changed to **True**.

- **Description**: (See Data Cell Access Security for information)
- **Action**: (See Data Cell Access Security for information)
- **Member Filters**: (See Data Cell Access Security for information)

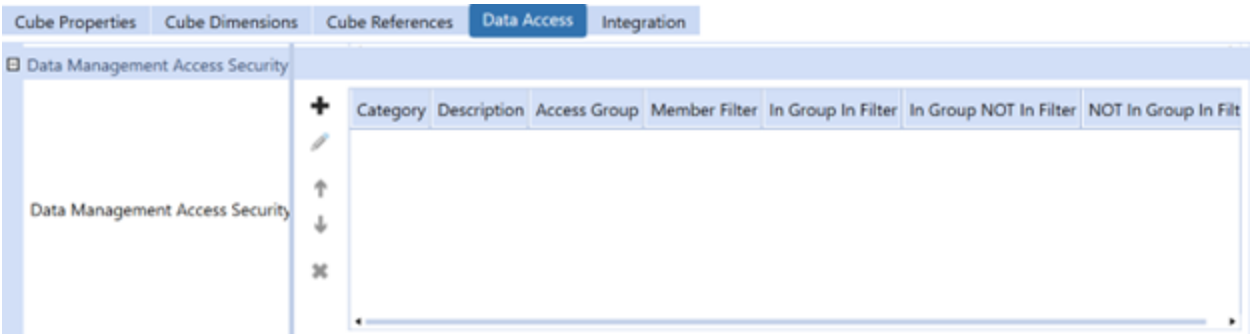
Data Management Access Security

This section provides a level of cube security when running processes from a data management sequence or step. See Data Management for more information.

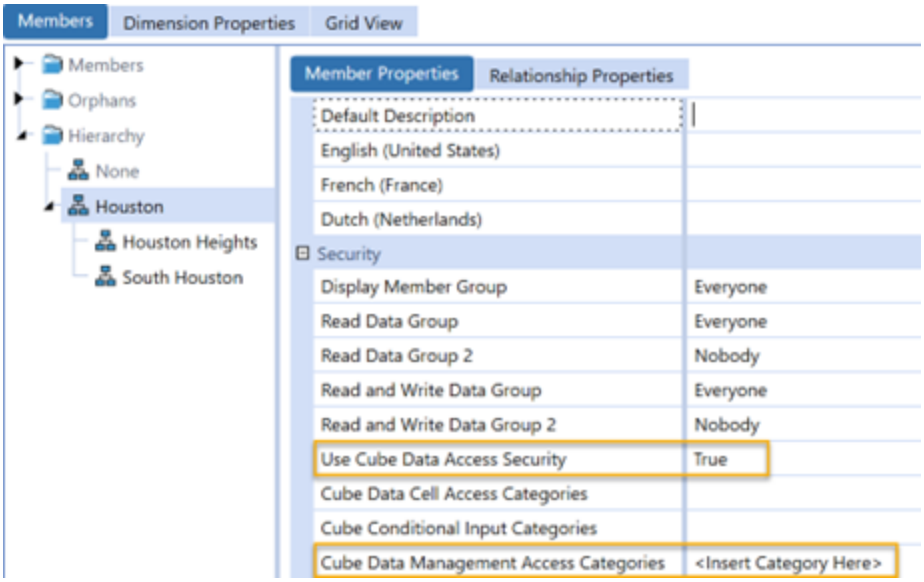
Data management access security shares the same properties and behaviors as data cell access security. The purpose of data management access security is to control who has access to read or modify cube data when using a data management process. This could be a data management process intended for running a data management sequence or step from the data management screen, workflow loading to the cube, or running a dashboard button with a data management sequence assigned to call a Finance business rule. As with these processes, they are focused on cube data unit or workflow data unit and not individual data cells. These are some of the examples where data management processes may require data management access security but not limited to only these.

Structure of Data Management Access Security

Category: This is an optional category name by which access rules can be named and grouped. The naming convention for category is free form and is limited to 100 characters. When a cube is created, by default, the data management access security section is blank.



If these categories are created, more than one can be applied to an entity's security settings. Enter an optional comma-separated list of category names that will be used when processing the cube's slice-level data access algorithms. Use empty text to process all categories. Include |Null| to represent nameless categories.



To add a category to **Cube Data Management Access Categories**, the **Use Cube Data Access Security** must be changed to **True**.

- **Description**: See Data Cell Access Security for information.
- **Action**: See Data Cell Access Security for information.
- **Member Filters:**The difference between Member Filters in data management access security and data cell access security or data cell conditional access security is Entity and Scenario dimension Types are the only options. The Member Filter focus is not at the data cell level, the data management access is data unit focused.

Integration

Integration settings control and determine what metadata is needed to structure and support the data flow through a workflow data loading process. This impacts workflow, data sources, transformation rules, and table structures in Stage and BI Blend. Integration settings are defined on a cube-by-cube basis.

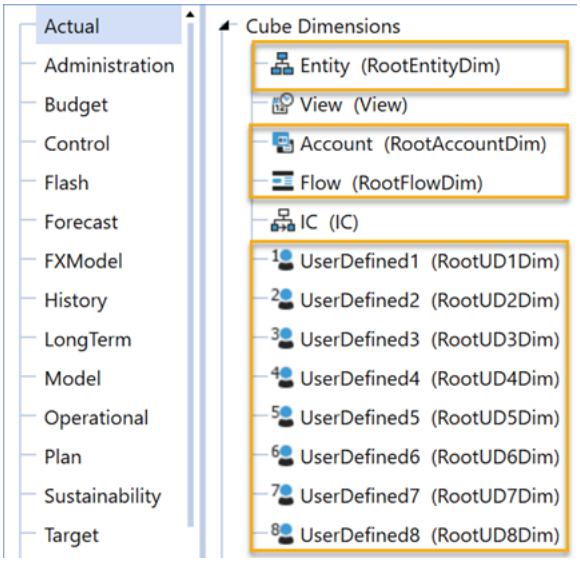
The Integration tab contains a list of all scenario types. With each type, four sections are available for configuration. This enables you to share the same integrations or configure independently. Dimensions and cube design drives what each scenario type has available. However, these four sections can be configured to meet the data loading and reporting demands by scenario type. The four sections are:

- Cube Dimensions
- Label Dimensions
- Attribute Dimensions
- Attribute Value Dimensions

Cube Dimensions

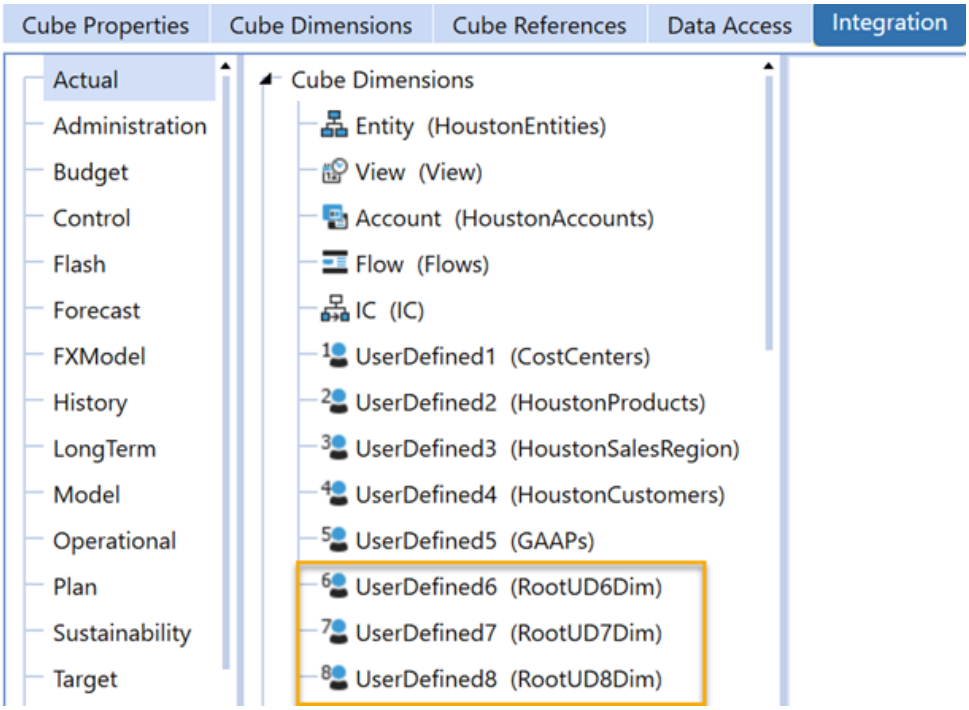
In this section, each scenario type isassigned dimension types defined in the Cube Dimensions tab.

When creating and configuring a new cube, each scenario type will have 13 dimension types enabled, and they will be populated with the RootXXXDim dimension for that dimension type.



Once dimensions are assigned to the dimension types in the Cube Dimensions tab in the cube, the dimension types by scenario type will populate with the appropriate dimension.

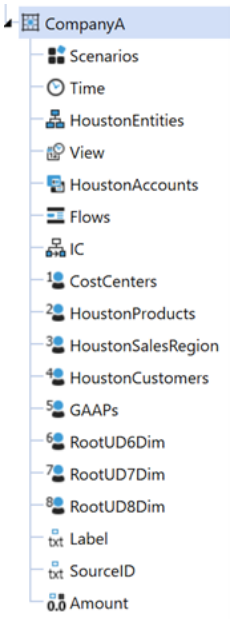
> **Example:** The example below shows the Integration tab after dimensions are assigned to dimension types by scenario type in the Cube Dimensions tab.



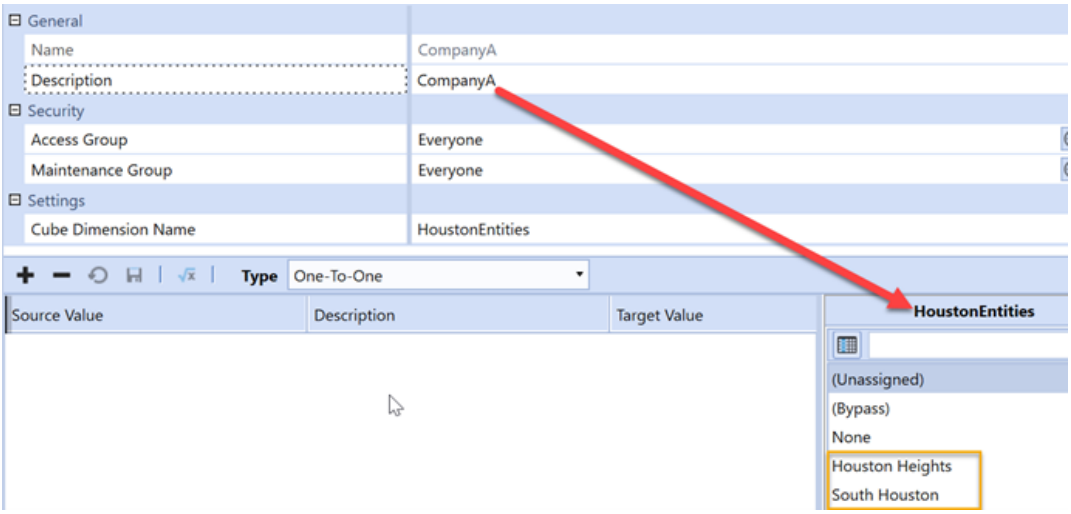
NOTE: All the dimension types are enabled, by default, in the Actual scenario type. This is apparent because none of the dimension types are grayed out. When the dimension type is disabled, the dimension type displays gray.

NOTE: In this configuration, UD6, UD7, and UD8 are not used. The RootXXXDim are assigned to those dimension types. This is an indication that the dimension types are not currently being used. Since RootXXXDim are assigned, they are configured correctly for future use.

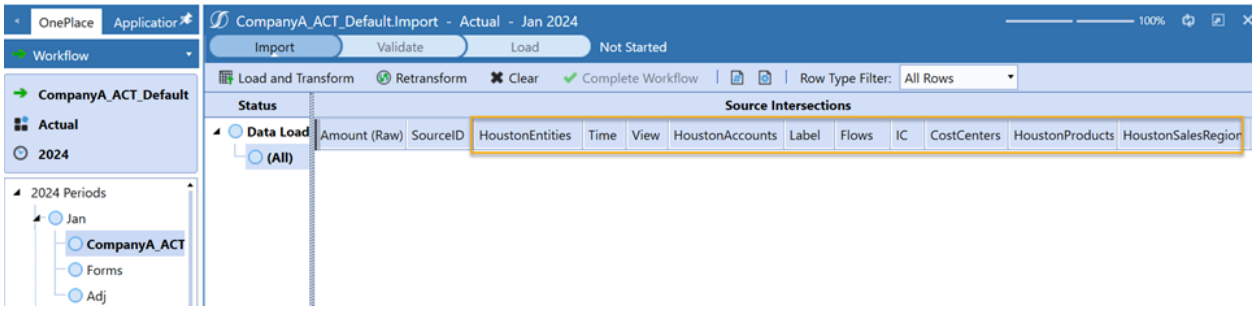
The configuration of the Integration tab is important as this drives everything data loading related. This impacts how the data sources are created, structured, and configured.



This impacts what dimension types, dimensions, and dimension members are used and available for the transformation rules. In this case, Entity = dimension type, HoustonEntities = dimension, while Houston and Houston Heights = dimension members.

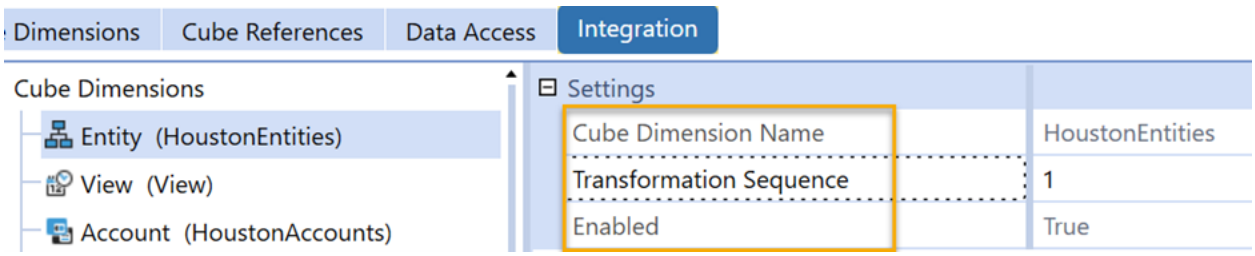


This controls how stage tables display through the import workflow process. Workflow profiles abide by the integration. The Integration tab integrates the data source, transformation rules, workflow profiles together.



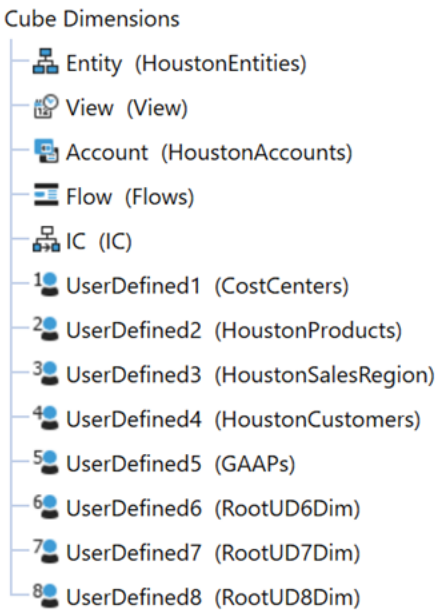
Cube Dimensions Settings

After dimensions are assigned to the dimension types by scenario type, the configuration will display in the Integration tab. Each dimension type has its own settings to control various behavior. When it is selected under cube dimensions in the Integration tab, a settings section will populate to the right. The settings section will contain three properties:



Cube Dimension Name: By default, the cube dimension name is grayed out and cannot be modified. The cube dimension name is driven by the dimension assigned to the dimension type for a scenario type. Any naming convention changes to the dimension name would be considered back in **Cubes** > **Dimensions** > **Dimension Library**. See Dimensions OverviewDimensions Overview

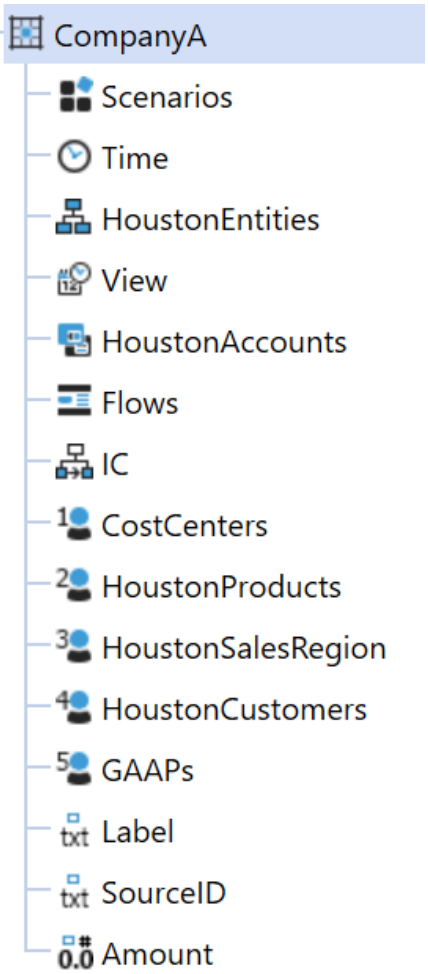
Transformation Sequence: By default, each dimension type is assigned a numeric value to identify the order in which the dimension type transformations should occur. By default, the order in which the transformations will occur is shown below Entity > UD8.



Enabled: By default, enabled is set to True for each dimension type for a scenario type. Entity, View, and Account Enabled properties are grayed out for every scenario type.

For any data loading process, entity, view, and account are the minimum dimension types needed to construct a workflow and a cube. Entity provides the dimension type for a cube data unit or unit of work. Entity can be defined as anything that is a driver for a well performing cube data unit such as legal entity, department, and more. Entity and Account combine for the workflow data unit. Both dimension types are required for integration. See Entity, Data Unit, and Workflow.

When a dimension type is not used for the scenario type, it is recommended that enabled is set to False. This way, when a data source and transformation rule is created, the dimension types that are not enabled, will not show up in the data source or transformation rules for mapping. The screenshot below shows the structure of a data source, at data source creation time, when dimension types UD5, UD6, and UD7 are not enabled.



In the case where the data source and transformation rules were created prior to setting the Enabled property from True to False for the unused dimension types, the data source will be constructed with all the dimension types. You would then have to manually delete the dimension types from the data source after the dimension types are disabled in the Integration tab of the cube.

When data sources and transformation rules are created and the unused dimension types are not removed, the unused dimension types must be mapped. Unused dimension types in the data source must be mapped to a column in the source file or source integration. Unused dimension types in the transformation rules must be mapped to the None dimension member, even though the dimension type is not being used and does not contain dimension members. When configuring the Integration tab by scenario type, always disable dimension types that are not being used for the scenario type. This reduces the configuration time for data sources and transformation rules and saves on maintenance.

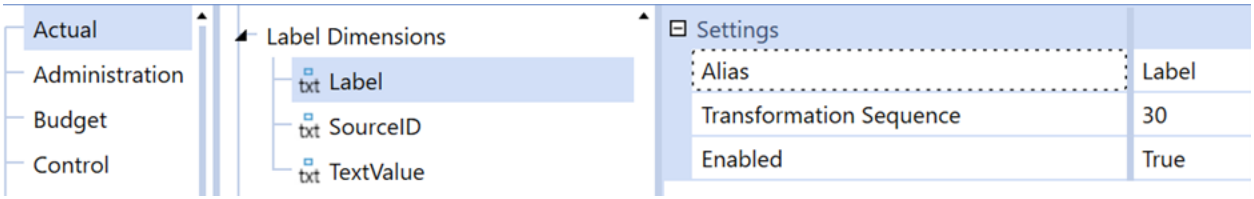
Label Dimensions

These are special fields that are helpful identifiers or additional text commentary during the data loading process. When defined in a data source, each one of these Label dimensions provides a unique purpose for each data record loaded into Stage.

The Label dimensions are defined as: Label, SourceID, and Text Value. Each one of the Label dimensions share the same settings for configuration: Alias, Transformation Sequence, Enabled.

The label settings are like the cube dimension settings. However, unlike cube dimensions, Label dimensions can have their own unique naming convention. The Alias property in the Label dimensions settings is a free form field. The max length of characters for the alias is 100.

The Label dimensions can also be configured by scenario type same as cube dimensions.



Label: This would be the description for any given account related to a data record from the source file or source integration. When Enabled is set to True for label, the Label dimension can be mapped in the data source to identify the description for the account. The label only exists in the stage table and does not get loaded into the cube.

Source ID: This is one of the keys to the data in the stage and should be enabled. It can be mapped in by a data source and set to a particular value in a file, the file name, or a tab name from an Excel file. Max length is 100 characters.

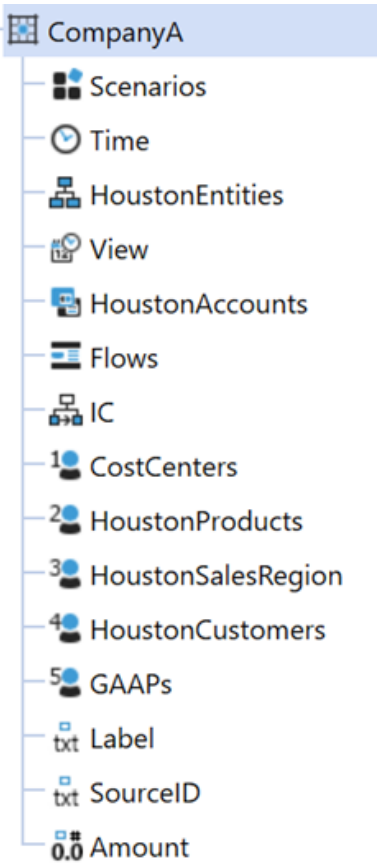
TextValue: This is to store large amounts of textual data. Max size is 2GB of text.

Attribute Dimensions

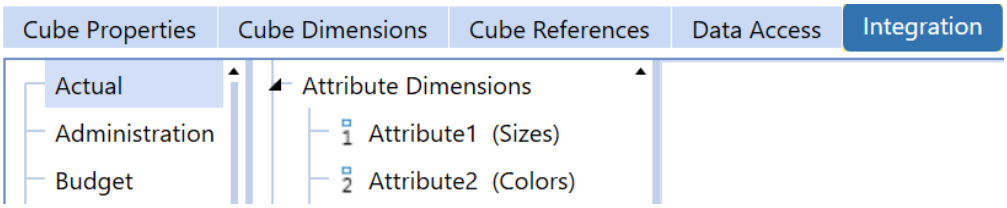
The Attribute dimensions are used independently or in concert with the cube dimensions for reporting purposes. When working with the cube dimensions, they can be considered as a feature or characteristic of a dimension and dimension member. As an alternative, they can also be created completely independent of cube dimensions. They can be used for operational purposes to report transactional level data that may have little to no alignment with any cube dimensions.

> **Example:** In this example, a golf manufacturing company produces golf clubs. Golf clubs may be a dimension member as part of a dimension type for summary financial reporting. However, there may be some operational reporting that needs additional information on the color of the golf clubs such as: white, black, or pink. These colors would be considered attributes and color be used in the Attribute dimensions.

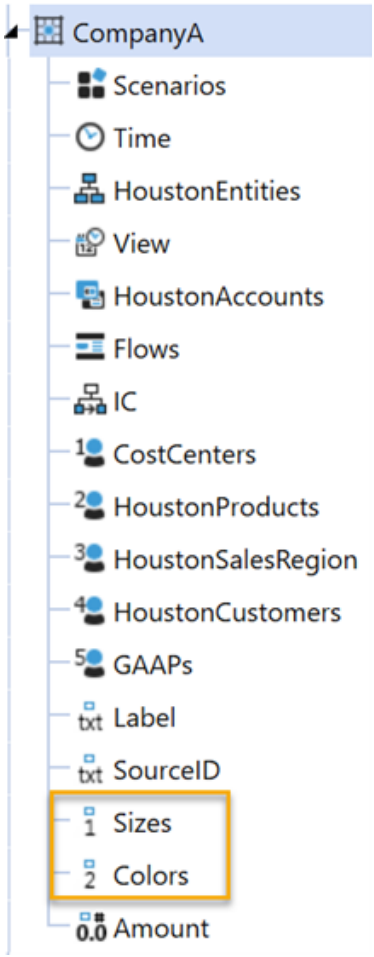
By default, Attribute dimensions are disabled. Each scenario type and process will determine whether they are needed. Therefore, the default behavior is to disable all the attributes. The same situation exists as with cube dimensions but in the opposite order of operation. By default, they are disabled compared to cube dimensions which are enabled. Attribute dimensions must be defined and configured prior to creating a data source. If they are not enabled at the time of data source creation, this is how the data source would appear as no attributes available but all cube dimensions available. See Cube Dimensions Settings Section for more information.



NOTE: By enabling the Attribute dimensions for a scenario type prior to the creation of a data source, this is how they will display:



NOTE: The Attribute dimensions are enabled for mapping within the data source, this is how the data source will display:



If the data source is created prior to setting the Enabled property from False to True for an Attribute dimension, the data source will be constructed with the dimension types configured and no Attribute dimension. You would then have to manually add the Attribute dimensions to the data source.

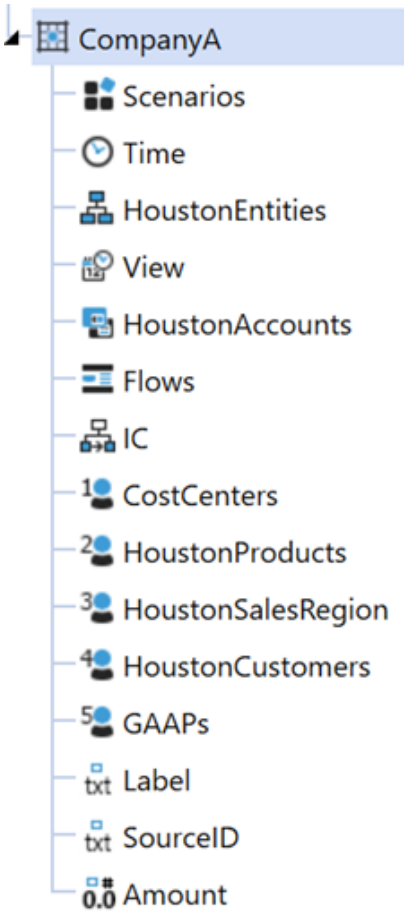
The Attribute dimension has 20 attributes available per scenario type. The attribute settings are like the cube dimension settings. However, unlike cube dimensions, the Attribute dimensions can have its own unique naming convention. The Alias property in the Attribute dimensions settings is a free form field. The max length of characters for the alias is 100.

Attribute Value Dimensions

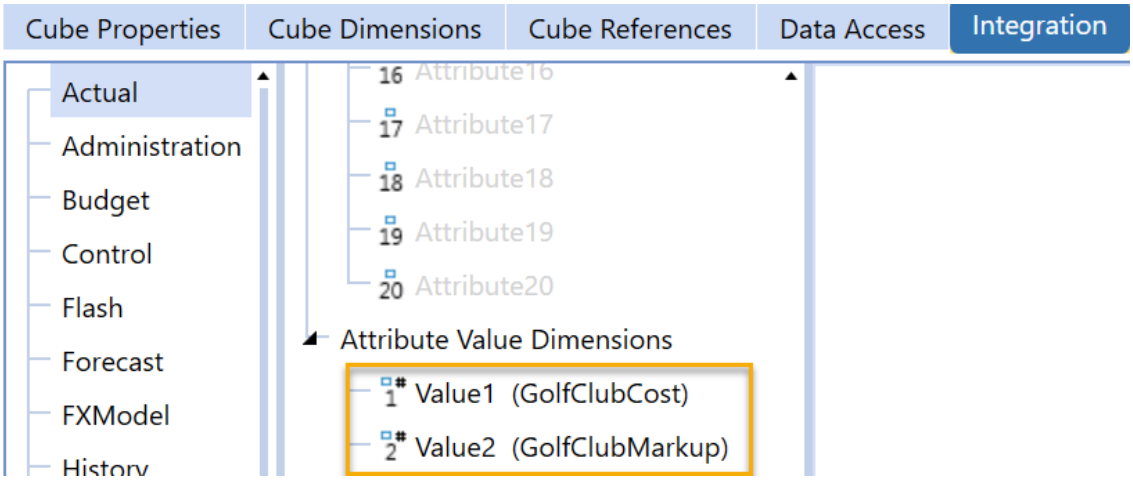
The Attribute Value dimensions are used to store a numeric value associated to an attribute member. There are 12 Attribute Value dimensions available and are defined as decimal data type and only numeric values can be stored. By default, Attribute Value dimensions are disabled. Each scenario type and process will determine whether they are needed. Therefore, the default behavior is to disable for all the values. The same situation exists as with Attribute dimensions. See Attribute Dimensions.



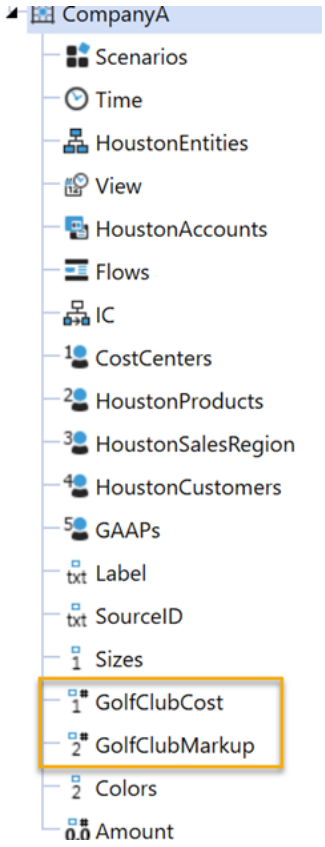
Attribute Value dimensions should be defined and configured prior to creating a data source. If they are not enabled at the time of data source creation, this is how the data source would display: no attributes values available but all cube dimensions available.



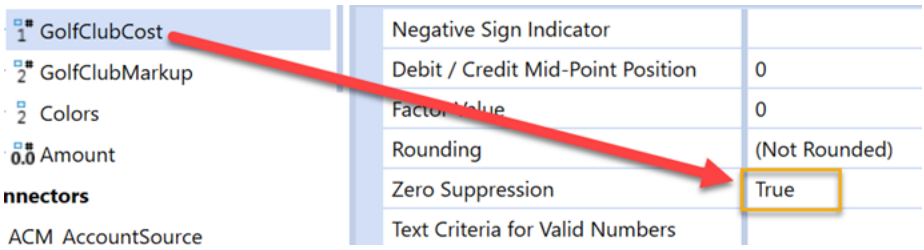
> **NOTE:** By enabling the Attribute Value dimensions for a scenario type prior to the creation of a data source, this is how they will display:



NOTE: The Attribute dimensions are enabled for mapping within the data source, this is how the data source will display:



When creating a data source with Attribute Value dimensions enabled, by default, the Zero Suppression setting for the Attribute Value dimension is set to True.



When developing a data source using Attribute Value dimensions where a specific value is 0, data may be suppressed. This can suppress the entire data row even if the amount field for Zero Suppression is set to False and has a value <> 0. The entire row will be suppressed in this case where Amount is <> 0 but the attribute value zero suppression is set to True and the value for attribute value is 0. The attribute value zero suppression will override the amount zero suppression.

Foreign Exchange Rates

Foreign exchange rates are stored in the system cube as a central repository for all other cubes to reference the currency rates. This reduces the need to place the rates in each cube as a default. Exchange rates give the system the ability to convert currency values in the database from one currency into another, for example, converting USD to Euro.

The settings in the Global Point of View essentially give a Cube View the currency rates that can be input or viewed in the system.

Grid Settings Intersection

FX Rate Type

Average Rate
The average currency rate of a period from the first day to the last day of the month.

Opening Rate
The currency rate at the beginning of the period.

Closing Rate
The currency exchange rate at the end of a period.

Historical Rate
The currency rate to be used for a specific historical Account calculation open for a specific special transaction to be valued on a specific date.

Time
Select time frame

Source Currency

Select type of currency

Destination Currency

Select destination currency for viewing in this Cube View

Rows and Columns

Row Axis

This drop-down menu defines what to display in the rows

Column Axis

This drop-down menu defines what to display in the columns.

Example of FX Rates:

Grid Settings	
Intersection	
FX Rate Type	AverageRate
Time	2013
Source Currency	USD
Destination Currency	USD
Rows And Columns	
Row Axis	Destination Currencies
Column Axis	Source Currencies

	CAD	EUR	GBP	USD
CAD				
EUR				
GBP				
USD				

Lock FX Rates

Use the Lock FX Rates feature to lock FX rate types. The feature prevents or allows all FX rate types rates to be changed. This may help eliminate mistakes and provides audit and task activity information around FX rate activities, adding integrity to the FX rate type data.

> **NOTE:** File Loads or XFSetRate functions will fail if an FX rate type is locked.

The application locks the FX rate type by time only, not by details of source or destination currency.

All locking is performed from the interface or a business rule.

Security Administrators have full rights and all functionality including locking and unlocking, by default. To assign non-administrators lock and unlock rights requires standard security group assignments in **Application** > **Security Roles**.

Non-administrators:

- Must be granted access to Application User Interface Roles/FxRatesPage. This will grant view access to the rates and the locking page.
- Access to the FX rate types, creating and deleting, and modifying/loading rates has not changed. Users must be granted application security roles/ManageFXRates.
- Users that will lock rates manually or withBRApi rules will require application security roles/LockFxRates.
- Users that will unlock rates manually or with BRApi Rules will require application security roles/UnLockFxRates.

Security Roles that Manage the Locking Features

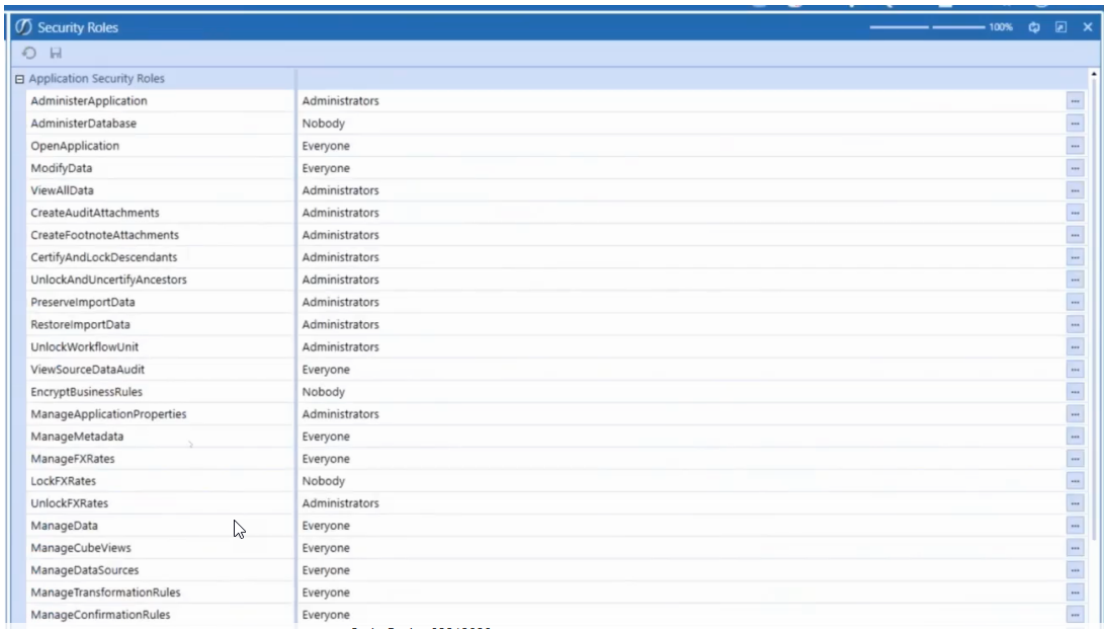
Depending on the settings for LockFXRates and UnlockFXRates, users will have access to the Manage FX Rate locking window.

The administrator user group by default has both roles. Other users can be assigned permissions for lock only, unlock only or both. You do not need to be able to manage the rates, lock/unlock is a separate role.

Users need security rights to each of these roles to lock and unlock.

To access FX Rates Security Roles

1. From the Application tab, go to **Tools** > **Security Roles**.



2. Change ManageFXRates to allow users visibility to the FX rates functionality.

3. (Optional) Change LockFXRates to allow users to lock rates.
4. (Optional) Change UnlockFXRates to allow users to unlock rates.

> **NOTE:** Users can have access to either LockFXRates or UnlockFXRates. They do not need access to both.

System Changes: Administrator Access to Lock FX Rates

- **Task Activity**: As the rates are locked and unlocked, the application tracks who performed the lock or unlock and when.
- **Audit FX Rate Lock Table**: Tracks the current status of the Rate Type and who performed the unlock or lock.

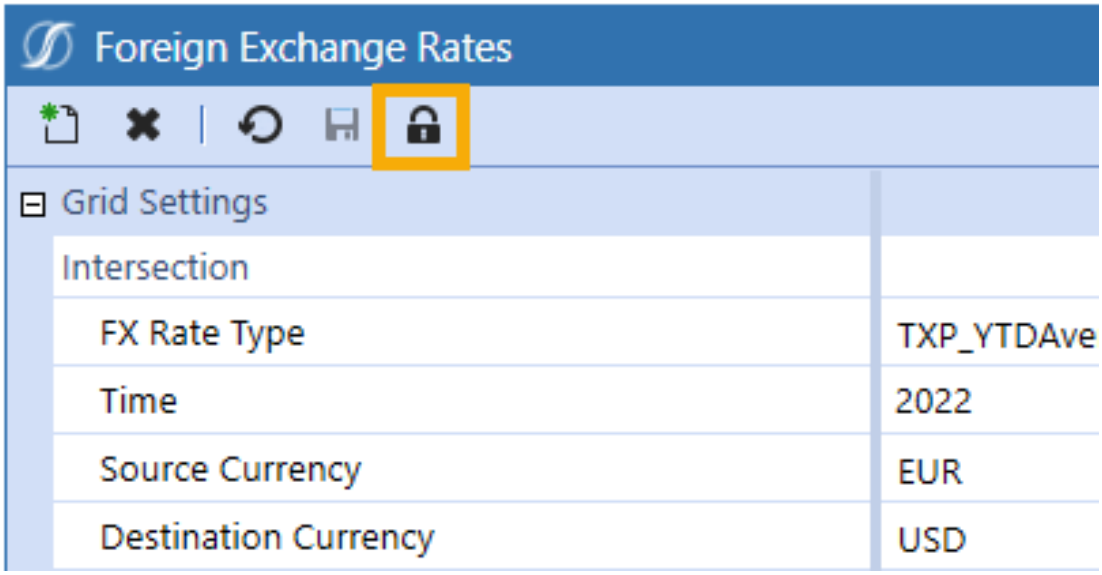
View Task Activity

There are two new task activity logs produced. Lock Fx Rate and Unlock Fx Rate will be generated whenever the lock status is modified.

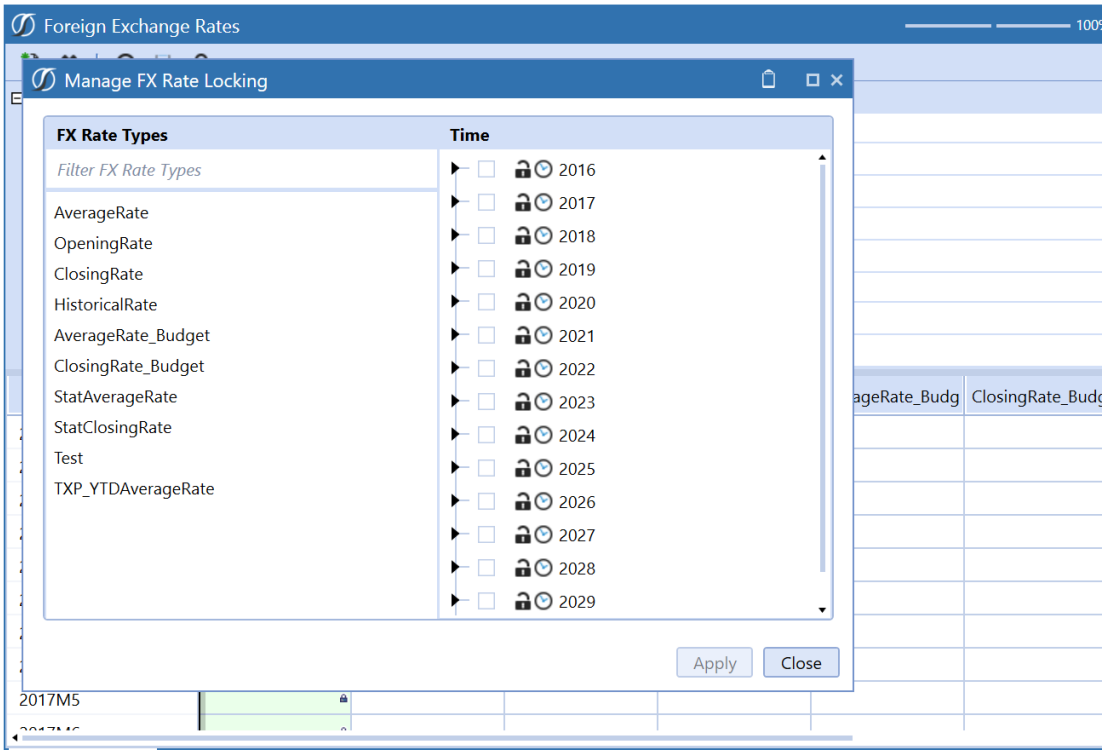
1. From the application, select **Task Activity**.
2. In the task list you will see which users have locked or unlocked rates and the time.

Lock and Unlock Rates

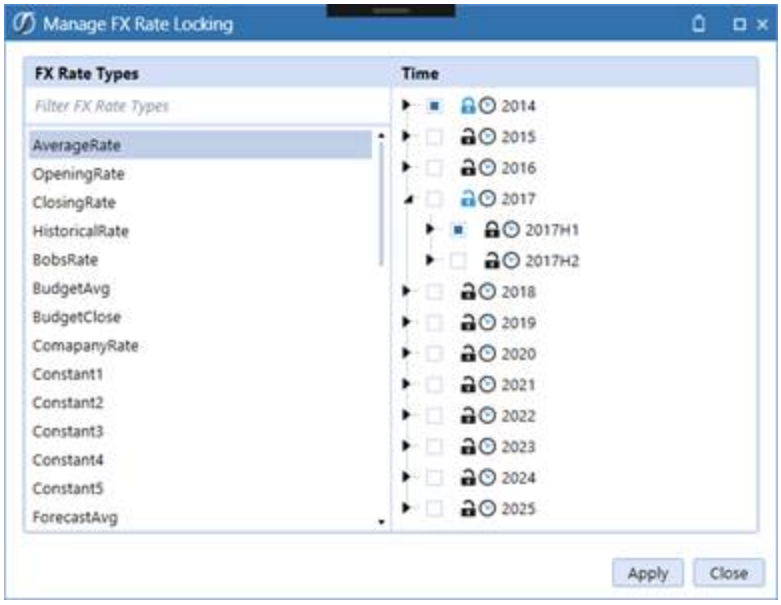
1. From the Application tab, go to **Cube** > **FX Rates**.
2. Select the lock icon at the top of the page.



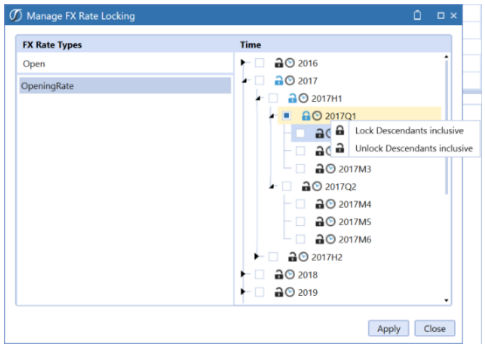
3. The Manage FX Rate Locking dialog box opens.



4. In **FX Rate Types**, select a rate type from the list or enter a type in the search box.
5. In **Time**, select a year or expand the year to view the full hierarchy.

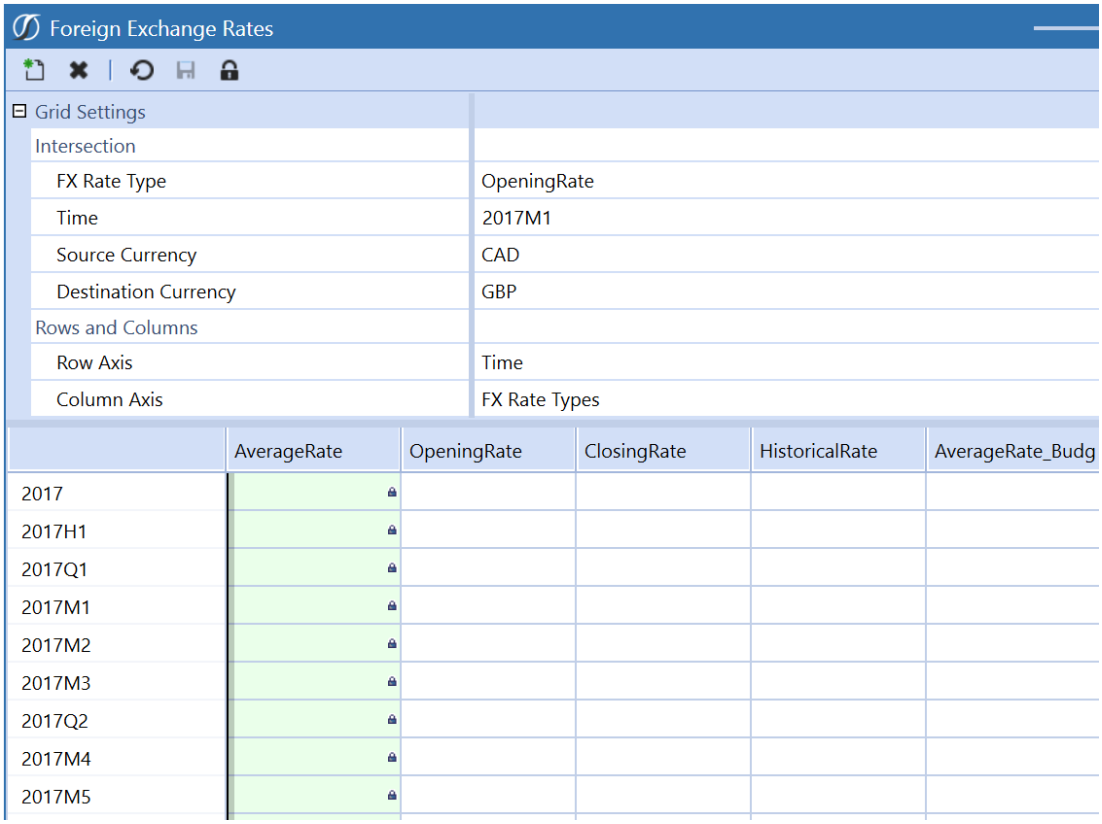


6. You can navigate the time structure using the expansion icons. If you select an item using the checkbox, it will set the lock status as a locked or unlocked padlock. Also, any lock that appears blue indicates that a descendant beneath it has a differing lock status.
    - If an item is Locked, the padlock is locked.
    - If an item is Unlocked, the padlock is unlocked.
    - If all the descendants are in the same lock state, the padlock is black.
    - If any of the descendants are in a different state, the padlock is blue.
7. You can lock an entire tree or lock or unlock specific items.
8. You can right-click to apply to a hierarchy and choose **Lock Descendants** inclusive to lock all within the time frame or **Unlock Descendants** inclusive to unlock all the within the time frame.



9. Select **Apply** and then select **Close**.

10. The FX Rates Grid is displayed. If the rate type and period is locked, you will see a green background and a lock icon in the corner of the cell.



Consolidation

Consolidation is the process of taking base level entities and aggregating them up a hierarchy to their parent. The hierarchy and add efficiencies where applicable, meaning some siblings may be able to run in parallel when they are processed. Calculations and translations are also run during this process.

Consolidations can be launched from the process step in Workflow, Cube Views or Forms. By triggering a consolidation, the existing POV is run for that time period and is consolidated for the respective period along with any of the prior periods for that year. For example, if a consolidation is triggered for June, it will consolidate January through June.

The following chart explains how the consolidation dimension works.

TOP - Contributes to the parent entity's Local member
Sum (SHARE + ELIMINATION + OWNERPOSTADJ)

OWNERPOSTADJ (Execute Data Unit Calculation Sequences, FinanceFunctionType = Calculate)

Stores Journal entries that are needed after the share and elimination calculations have been performed for an entity and parent.

ELIMINATION (Execute DUCS, FinanceFunctionType = Calculate)

Elimination algorithms (Use default or implement FinanceFunctionType.ConsolidateElimination in a BR)

Start with SHARE Data Unit and eliminate data cells where IC is not None. If a common parent entity was reached for both the entity and the IC entity, generate an offset amount and a plug account amount.

SHARE (Execute DUCS, FinanceFunctionType = Calculate, only if not using default calc-on-the-fly)

Default Share is calculated on-the-fly

Sum (TRANSLATED + OWNERPREADJ) * % Consolidation

Share algorithms (Use default or stored, or implement FinanceFunctionType.ConsolidateShare in a BR)

OWNERPREADJ (Execute DUCS, FinanceFunctionType = Calculate)

Stores Journal entries that are needed before the share and elimination calculations have been performed. This is a relationship-level member, so the data is for a specific parent entity.

TRANSLATED (Execute DUCS, FinanceFunctionType = Calculate)

The Translated member is a shortcut to the Consolidation member that matches the parent entity's default currency (e.g., USD). It stores data that was generated by multiplying local currency data by FX rates, and then executing the DUCS.

Translation algorithms (Use default or implement FinanceFunctionType.Translate in a BR)

Determine FX rates and multiply to create a translated data unit from the local currency's data unit.

LOCAL (Execute DUCS, FinanceFunctionType = Calculate)
The Local member is a shortcut to the Consolidation member for the entity's default currency (e.g., EUR). It stores input data in base entities, or data that was consolidated by combining data from the TOP consolidation member from lower-level child entities.

Example of a Consolidation

Entity	Currency
GolfStream	USD
Clubs	USD
Frankfurt	Euro
Houston	USD

Clubs + GolfStream Relationship	Clubs	Interco AR	**Top**	70	
	Clubs	Interco AR	OwnerPostAdj	0	
	Clubs	Interco AR	Elimination	0	
	Clubs	Interco AR	**Share**	70	
	Clubs	Interco AR	OwnerPreAdj	0	
	Clubs	Interco AR	**Translated**	70	
	Clubs	Interco AR	Local	70	Pulled from Frankfurt Child Top (& other child entities)
Frankfurt + Clubs Relationship	Frankfurt	Interco AR	**Top**	**70**	Sum of Share + Elim + OwnerPostAdj
	Frankfurt	Interco AR	OwnerPostAdj	0	Calculation Sequence
	Frankfurt	Interco AR	Elimination	-50	Elimination: Consolidation Logic
	Frankfurt	Interco AR	**Share**	**120**	Share: Consolidation Logic (at 100%)
	Frankfurt	Interco AR	OwnerPreAdj	0	Calculation Sequence
	Frankfurt	Interco AR	**Translated (USD)**	**120**	Translation Logic then Calculation Sequence
	Frankfurt	Interco AR	Local (EUR)	100	Calculation Sequence

Launching a Consolidation

In Workflow Profiles, set up calculation definitions to process the appropriate calculation, translation, or consolidation type. Right-click on the appropriate cell in a Cube View or form to view process options:



Calculate

Runs calculations at the entity level within the Local member of the Consolidation dimension without translating or consolidating.

- Calculate
- Force Calculate
- Calculate with Logging
- Force Calculate with Logging



Translate

Runs the calculate step above at the entity and then translates data within the translated member of the Consolidation dimension for each applicable relationship.

- Translate
- Force Translate
- Translate with Logging
- Force Translate with Logging



Consolidate

Runs the calculate and translate steps and then completes the calculations required all the way up the Consolidation dimension.

- Consolidate
- Force Consolidate
- Consolidate with Logging
- Force Consolidate with Logging

Force menu items such as Force Consolidate will run as if every cell included is marked as requiring calculation, translation or consolidation.

Consolidate and Force Consolidate check and determine a parent member's calculation status and all children of the parent before consolidating any data. The difference is Consolidate checks calculation status and, if the status is OK, it accepts it and continues the consolidation process. Force Consolidate runs as if every member needs to be consolidated regardless of its actual calculation status and does not bother querying calculation status. While they both perform optimally, there are some cases where one performs better than the other. See the examples below to learn more about when to use Force Consolidate rather than Consolidate.

Calculation Status State 1: Actual Scenario, Loading for December



In this case, Consolidate will perform better than Force Consolidate because every month prior to December has an OK calculation status. This means the data for that month has not changed since the last consolidation and only the month of December needs to be consolidated. If a Force Consolidate was used, every calculation would be performed again whether it is necessary or not, therefore taking longer in the consolidation process.

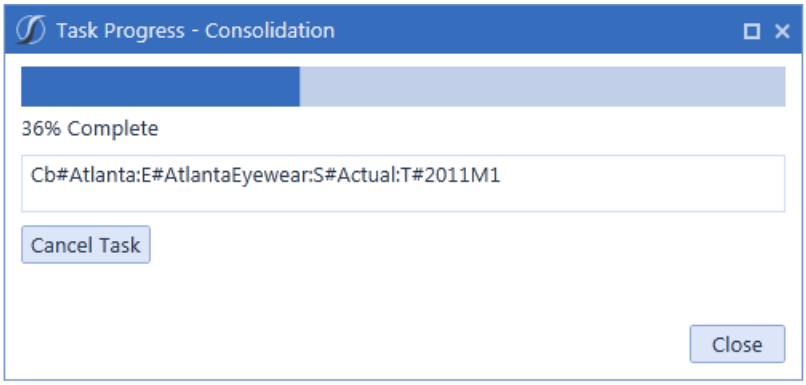
Calculation Status State 2: Plan Scenario, Loading for 12 Months



In this case, Force Consolidate will perform better than Consolidate because every month needs to be consolidated. If a Consolidate was used, the system would needlessly check each calculation status before calculating each period. A Force Consolidate will calculate all periods regardless. See **Calculation Status** for details on the status codes.

> **NOTE:** Forced calculations that run on base entities from open periods will not impact the status of parent entities for closed periods if the data is in an OK state. If a metadata change occurred and entities are in an OK, MC state, all periods and entities will recalculate, regardless of Workflow open/close state.

The logging items (such as Force Translation with Logging) trigger additional detailed logging which can be viewed in the Task Activity area. Drill into a log to see the length of time and details about every calculation. A progress window displays consolidations.



Calculation Status

If the existing data set for a POV changes, the calculation status is updated accordingly. A timestamp table is used, and servers are synchronized to ensure the calculation status is always accurate.

To explore calculation status, build a Cube View where the view member in the POV is set to CS, the columns are the time periods, and the rows are set to the desired entity structure. For example:

	Jan 2011	Feb 2011	Mar 2011
All Orgs	CN	CN	CN
Total GolfStream	CN	CN	CN
Clubs	CN	CN	CN
NA Clubs	CN	CN	CN
Canada Clubs	OK,MC	OK,MC	OK,MC
Montreal	OK,MC	OK,MC	OK,MC
Quebec City	OK,MC	OK,MC	OK,MC
US Clubs	CN	CN	CN
Augusta	OK,MC	OK,MC	OK,MC
Carlsbad	OK,MC	OK,MC	OK,MC
Houston	CN	CN	CN
Houston Heights	OK	OK	OK
South Houston	OK,MC	OK,MC	OK,MC
Europe Clubs	CN	CN	CN
Frankfurt	OK,MC	OK,MC	OK,MC
Golf Balls	OK,MC	OK,MC	OK,MC

These status codes indicate what action to perform:

- **OK**: The data for this intersection has not changed since the last calculation
- **OK, NA**: Calculation not required. No data
- **OK, MC**: The intersection was calculated but metadata changed due to modifications to artifacts such as business rules associated with this cube, formulas and FX rates. This is a clue that if the calculation is run again, the results may not be the same.
- **CA**: Calculate data since an import was run or data was entered
- **TR**: Translate
- **CN**: Consolidate because data for a child entity has changed
- **CA, CN**: Calculate and consolidate
- **CA, TR**: Calculate and translate
- **TR, CN**: Translate and consolidate
- **CA, TR, CN**: Calculate, translate and consolidate

Currency Translation

Currency can be converted from one currency to another. This process utilizes the defined FX rates in the FX rates portion of the product. See Foreign Exchange Rates.

The base entity can be converted to the parent entity's currency if required. The currency translation is run as needed based on the configuration of the parent/child currencies as stored in the Entity dimension. This feature can be run independently of a consolidation if required. Right-click in the appropriate cell and the translation option will display.



The Consolidation dimension is shown above. For example, if an entity's currency is Euro, and data is written to the EUR member, that value is also displayed in the Local member. The Local member is a pointer to the appropriate local currency member where the data is stored. If that European entity in the image above is consolidated to a company in the UK that has GBP as their currency setting under entities, when the European entity is consolidated into the UK parent, the European entity's Translated member will reference the translated value which is stored under the GBP member under Currencies.

Calculation status determines when data was entered. If the Local currency's calculation status is CA because the data has not been calculated yet, then the foreign currencies would also need to be translated and calculated resulting in a TR calc status. If a calculation is done on the Local currency in order to make its calc status OK, the foreign currencies are still going to be TR because that data has not been translated or calculated yet. Also, if a foreign currency has an OK calculation status, and a foreign currency journal is entered, that currency then becomes TR.

How the Origin Dimension Works with the Consolidation Dimension

The Origin and Consolidation dimensions are separate but work well together. Origin primarily calls out how a number originated. This section explains how each member of the Origin dimension interacts with members of the Consolidation dimension.

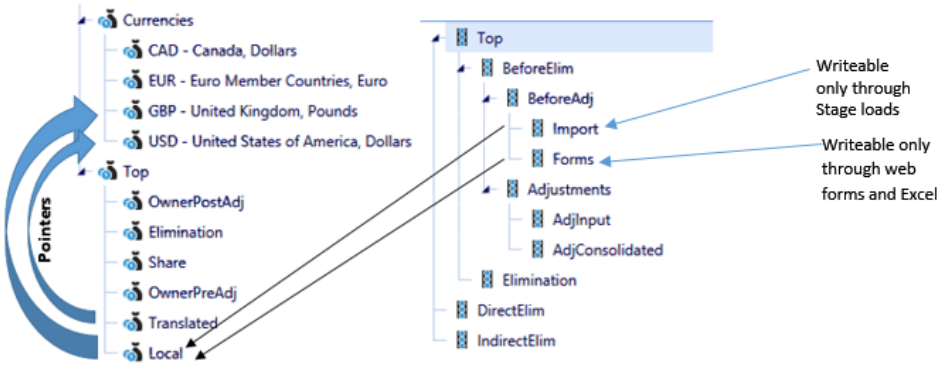
The consolidation process starts at Local, which is the same data as one of the currencies under the Currencies member, based on the entity's default currency setting. If the entity is base-level, the data can be loaded into the Local consolidation member using the Import Origin member, and data can be typed into the Local consolidation member using the Forms Origin member. If it is a parent entity, the Local consolidation member is read only (except for Journals) as it represents data that has been rolled up from child entities (using the Import, Forms, and the AdjConsolidated Origin members).

Regardless of whether the entity is base-level or a parent, Journals can be entered using the Local, Translated, OwnerPreAdj, OwnerPostAdj, and any Currency consolidation member. For all the consolidation members, Journals are always posted to the AdjInput Origin Member. As data is rolled up from child entities into a parent entity during the consolidation process, the AdjConsolidated Origin member will contain the combined values from the child entities' Journals.

Imports and Forms

When data is imported through Workflow to the data stage, it is happening through the Import member in the Origin dimension. When data is entered in web or Excel Forms, it is happening in the Forms member in the Origin dimension (with one exception discussed under Adjustments). The Import and Forms members are coupled with the Local member of the Consolidation dimension.

Consolidation Dimension Origin Dimension



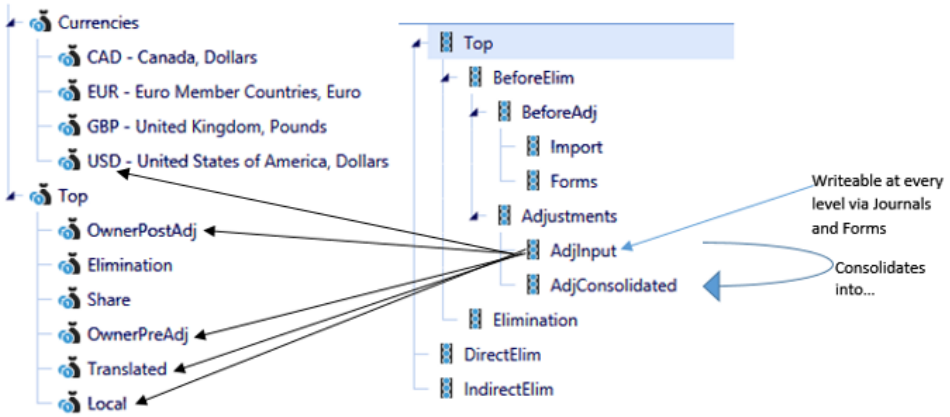
Adjustments

Adjustments are created either through Journal entries, or in special instances, through Forms that do their input as a Journal would. Both do their input into the AdjInput member of the Origin dimension. Adjustments can be made to the following members of the Consolidation dimension: Local, Translated, OwnerPreAdj and OwnerPostAdj.

When a consolidation is run, the AdjInput entries in child entities are consolidated into the AdjConsolidated members in the parent.

Users can drill down into the Adjustments member in a parent to show adjustments made in both the parent and child entities.

Consolidation Dimension Origin Dimension

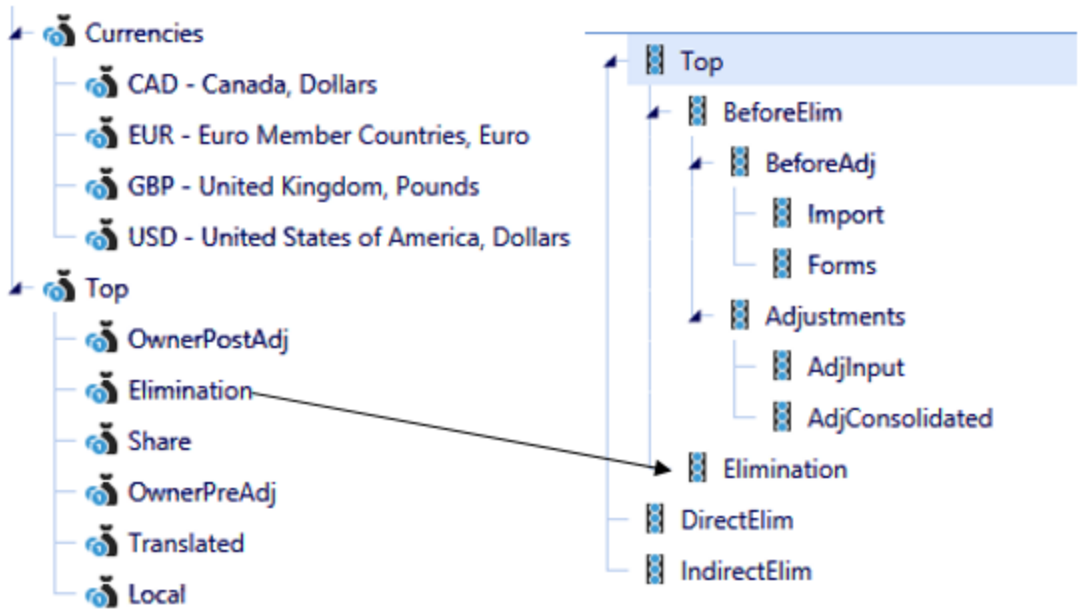


Eliminations

When eliminations are calculated, the Elimination member of both the Consolidation and Origin dimensions is updated. The primary purpose of the Elimination member within the Origin dimension is to allow visibility from the top down without those figures getting lost during the consolidation process.

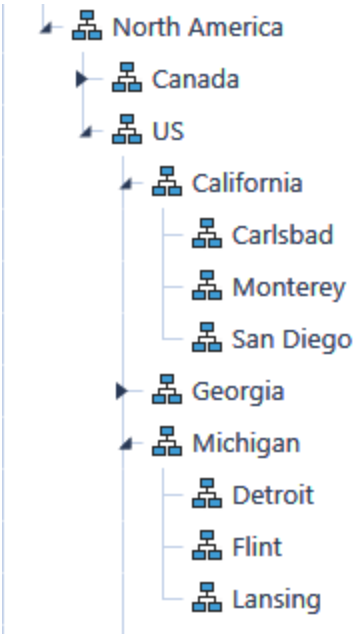
Users can drill down into the O#Elimination member in a parent to show the elimination entries made in both the parent and child entities.

Consolidation Dimension Origin Dimension



Intercompany Elimination

Intercompany elimination is the process of canceling out account balances for intercompany partners for intercompany accounts with any unresolved balance being placed in a plug account.

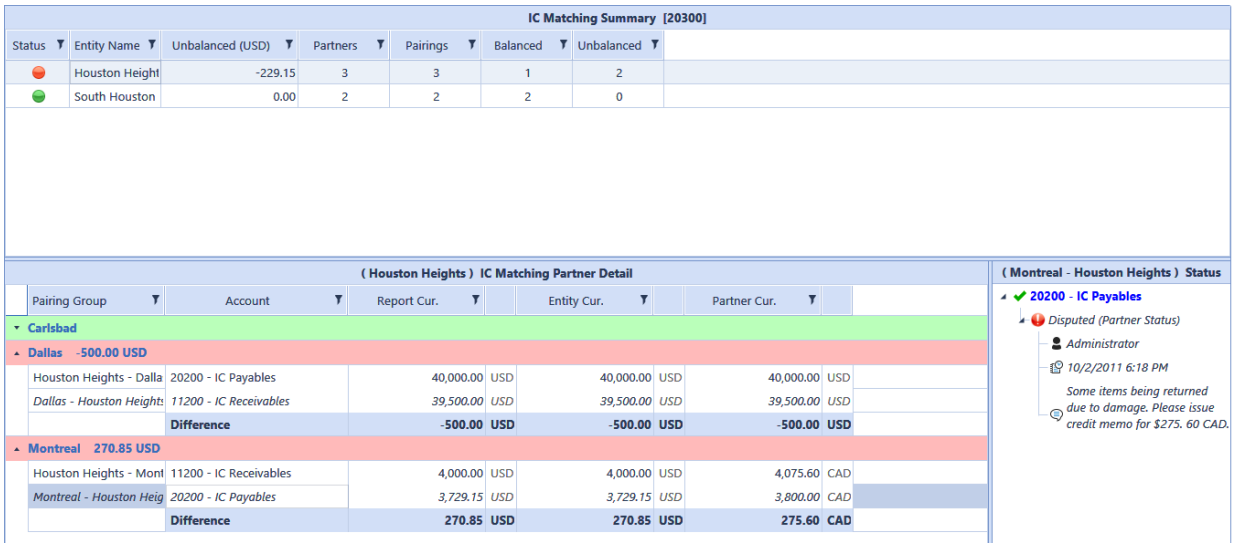


The entity structure above belongs to GolfStream, a fictitious golf manufacturer. If the Detroit entity sold golf club shafts to Monterey who assembles the final club product, this would be an intercompany transaction. The following prerequisites must exist for the transaction to eliminate:

- Monterey and Detroit must have their Is IC entity property set to True.
- The account's intercompany receivables and intercompany payables must be set with the Is IC account property set to True and the plug account pointed to a third account.
- The intercompany entries must properly note the intercompany partner in the IC dimension member. For example, Detroit would book an entry to intercompany receivables and the IC member for that entry would be Monterey.

Intercompany eliminations occur once the values roll up to a common parent. As the consolidation begins, Detroit consolidates its values to Michigan and Monterey consolidates its values to California. An elimination does not occur because they have not yet consolidated their values to a common parent. The elimination occurs when Michigan and California are consolidated into the US common parent. The two intercompany values will be eliminated at this level with any discrepancies being posted to the plug account.

In another example in workflow, Houston Heights and South Houston trade with Carlsbad, Dallas, and Montreal. In the screen shot below, South Houston is shown in green because it is balanced within an acceptable tolerance of $1. The $0.59 discrepancy is booked to the related plug account.



IC Matching Summary [20300]

Status	Entity Name	Unbalanced (USD)	Partners	Pairings	Balanced	Unbalanced
	Houston Height	-229.15	3	3	1	2
	South Houston	0.00	2	2	2	0

(Houston Heights) IC Matching Partner Detail

Pairing Group	Account	Report Cur.		Entity Cur.		Partner Cur.	
Carlsbad							
Dallas -500.00 USD							
Houston Heights - Dalla	20200 - IC Payables	40,000.00	USD	40,000.00	USD	40,000.00	USD
Dallas - Houston Heights	*11200 - IC Receivables*	*39,500.00*	*USD*	*39,500.00*	*USD*	*39,500.00*	*USD*
	Difference	**-500.00**	**USD**	**-500.00**	**USD**	**-500.00**	**USD**
Montreal 270.85 USD							
Houston Heights - Mont	11200 - IC Receivables	4,000.00	USD	4,000.00	USD	4,075.60	CAD
Montreal - Houston Heig	*20200 - IC Payables*	*3,729.15*	*USD*	*3,729.15*	*USD*	*3,800.00*	*CAD*
	Difference	**270.85**	**USD**	**270.85**	**USD**	**275.60**	**CAD**

(Montreal - Houston Heights) Status

- 20200 - IC Payables
  - *Disputed (Partner Status)*
    - *Administrator*
    - *10/2/2011 6:18 PM*
    - *Some items being returned due to damage. Please issue credit memo for $275. 60 CAD.*

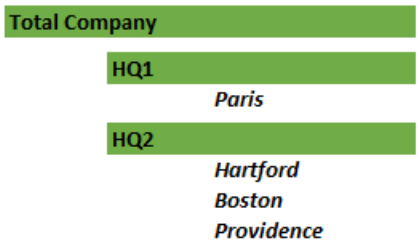
Houston Heights is shown in red because there is a $229.51 discrepancy. Details on the discrepancy are shown in the lower part of the screen. Houston Heights' discrepancy with Montreal is shown at the bottom in the application reporting currency (USD), Houston Heights' currency (USD), and the partner currency for Montreal (CAD). Through the right-click menu, leave a status and description that each partner can see. As these values roll up the entity structure, the parent entities can also see this detail.

See "Workflow Profiles" in the *Workflow Guide* for more details on how to set up intercompany matching with workflow.

> **NOTE:** Using Extensible Dimensionality to extend accounts used within the intercompany matching process is not a recommended practice.

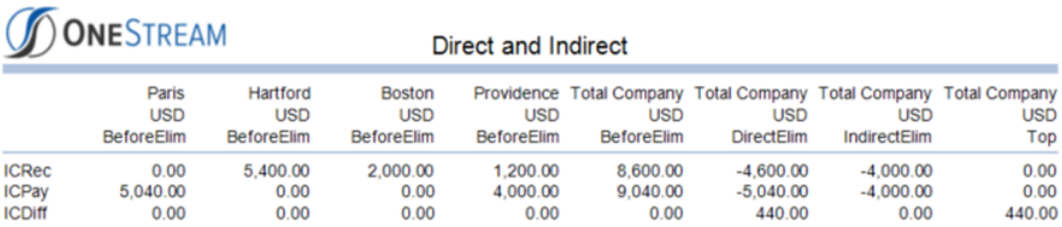
Direct and Indirect Eliminations

The Origin dimension contains the Direct and Indirect members to simplify reporting on eliminations. The Direct member returns the results of eliminations that occur from transactions, removing the direct children of a parent entity. Indirect returns the total eliminations that occurred outside the direct children of a parent entity.



In the example, transactions that occurred between the HQ2 entities eliminate at HQ2. Transactions between members of HQ1 and HQ2, such as a transaction between Paris and Hartford, eliminate at the first common parent, Total Company.

Reporting on the results at the Total Company level, Direct returns results that occurred between the HQ1 and HQ2 groups. Indirect at Total Company would allow reporting on eliminations outside its direct children, HQ2 eliminations.



Direct and Indirect

	Paris USD BeforeElim	Hartford USD BeforeElim	Boston USD BeforeElim	Providence USD BeforeElim	Total Company USD BeforeElim	Total Company USD DirectElim	Total Company USD IndirectElim	Total Company USD Top
ICRec	0.00	5,400.00	2,000.00	1,200.00	8,600.00	-4,600.00	-4,000.00	0.00
ICPay	5,040.00	0.00	0.00	4,000.00	9,040.00	-5,040.00	-4,000.00	0.00
ICDiff	0.00	0.00	0.00	0.00	0.00	440.00	0.00	440.00

Custom Consolidation

If the standard consolidation or translation logic does not meet your project requirements, deploy custom business rules. The first step in this process is defining the cube's properties settings. See Cubesfor Consolidation Algorithm Type and Translation Algorithm Type settings. The default setting is standard but you can change this to custom for additional flexibility.

Next, write a Finance business rule. See:

- "Application Tools" in the *Application Tools Guide* for applying business rule logic by finance function type to discover what you can customize.
- "Formulas" in the *Application Tools Guide* for common customizations
- *The API Overview Guide*

Reference the Finance business rule under any of the Business Rule 1-8 properties for each cube on which to use the logic. Also see Data Unit Calculation Sequences (DUCS).

Matrix Consolidation: Eliminating Beyond Legal Entity

Matrix consolidation is a term commonly used when finance teams want to prepare their management and statutory financials concurrently. This prevents the need to maintain separate scenarios and processes in the system. It will usually involve running eliminations on something more than a legal entity.

In OneStream, this can mean using a User Defined (UD) dimension as part of the elimination process.

A common use case is to run eliminations between profit centers (PCs) or segments. In this example, inter-profit center eliminations will be used, but that is not the only potential use case. The need is for an elimination to occur only at the first common parent in both the legal entity and profit center hierarchies.

Entity Dimension or User Defined Dimension

There is a requirement to do eliminations on a level of detail below the legal entity/company code level. The example in this section is to generate eliminations between profit centers. The main options are Entity dimension or User Defined dimension, which are described in the following sections.

Entity Dimension Option

Include this profit center detail in your Entity dimension as base members. These will be children of the legal entity members.

	Entity Dimension Option	
	Pros	**Cons**
Business Logic	No additional logic required to get eliminations running by PC	• Impacts consolidation performance more than the UD dimension option in a typical setup. This is due to the multiplication of members in the Entity dimension, more data units, to be consolidated. Analyze the impact for each project. • If you move a PC in the entity hierarchy, you will need to reconsolidate all history.

	Entity Dimension Option	
	Pros	**Cons**
Dimensions	• Uses fewer UD dimensions than the UD dimension option • Generally, only appropriate when PCs are unique to entities. Otherwise, they would need to be duplicated for each entity.	• Can result in a large Entity dimension • To achieve some reporting, alternative entity hierarchies and additional consolidations may be required. • Often leads to creation of additional artificial or journal PC entities to contain data that does not need to be captured by PC, for example, balance sheet data, which creates more entities that need to be consolidated • When PCs are not unique to entities, they create many duplicate entities. This should be avoided. • Less flexible because PCs need to be created and moved within the entity hierarchy

	Entity Dimension Option	
	Pros	**Cons**
Workflows	If the responsibility structure and workflow design are by PC, the workflow design and build can be improved.	• If the responsibility structure and workflow design are by PC, this option may make workflow design simpler. • Makes PCs the basis for everything where data is stored, processed, and locked
Reporting and Matching	• This may be a familiar approach, aligning with the reporting and matching in legacy systems. • Standard intercompany (IC) matching reports work for PC matching, although this requirement is less common.	• Out-of-the-box matching is only at PC level. Legal entity matching requires custom reporting. • Alternative entity hierarchies and consolidations may be required to achieve some reporting. • To see legal entity level data, as they will be parent entities, it is required to run consolidations.

	Entity Dimension Option	
	Pros	**Cons**
Security	Native using the Entity dimension	Requires maintenance on a PC level even if not required on that level

User Defined Dimension Option

Include this profit center detail in a User Defined dimension.

	User Defined Dimension Option	
	Pros	**Cons**
Business Logic	• Logic can be customized to specific requirements. • Does not add additional members, data units, to the Entity dimension, which is beneficial for consolidation performance in a typical setup • Running a consolidation will run a statutory and management consolidation in parallel.	Requires additional development time for business logic if it is not part of a starter kit

	User Defined Dimension Option	
	Pros	**Cons**
Dimensions	• A cleaner Entity dimension to support legal entity and group reporting • Matrix view of consolidation can be created, for example, with entities in rows and PCs in columns • Can be combined with extensibility if PCs are not applicable to all entities and divisions	Requires the use of multiple UD dimensions (one for PC and another for PC counterparty). See UD7 - New Counterparty Profit Center.
Workflows	Often more closely aligns with the responsibility structure for actuals (by legal entity)	None identified
Reporting and Matching	• Standard IC matching reports will support legal entity matching. • Consolidation is not required to view total legal entity values, pre-elimination data.	Custom IC matching reports may be required for PC matching. This is a less common requirement.

	User Defined Dimension Option	
	Pros	Cons
Security	Native using the Entity dimension if security is driven by legal entity	Requires slice security, Cube Data Access security, if required at PC level, which can impact reporting performance if security is complex

Other Design Considerations

- **Data Quality of Matrix Counterpart**: All intercompany data from the source system needs to be sourced for all matrix dimensions. It will negatively impact user experience if this data is not readily and accurately available in the source system. Significant manual input is required.
- **Stability of the Matrix Dimension**: Consider whether the profit center hierarchy will change regularly as relationships change. This requires significant consideration in the design phase. See Org-by-period in the User Defined Dimension.
- **New or Existing Application**: The choice of solution may depend on whether this is a new implementation or an addition to an existing one. It will likely be easier to add a new User Defined dimension to an existing application rather than redevelop the Entity dimension.
- **Performance**: Common design considerations of performance, data unit sizes, and number of data units apply. In addition, because of the nature of matrix consolidation, it is unlikely that the Matrix dimension can be extended because it is required for the elimination postings. Therefore, carefully consider the size and sparsity of this dimension when thinking about the performance of your design.

- **Elimination or Matching**: If eliminations happen on profit center, it does not mean that month end intercompany matching must be at this level. It is important to clarify these as separate requirements.
- **Workflow**: Consider the responsibility structure of the organization. It will have a big impact on the decision. If the true process (loading, locking, calculating, and certifying the data) is by profit center, then this could be a good justification for using the Entity dimension. However, it is more typical that these are based on legal entity for actuals, which makes a User Defined dimension solution more appropriate.

Option Overview

The best approach varies depending on the specific requirements. Review the benefits and considerations of each approach. Adding members to the Entity dimension creates additional overhead during consolidation because the system must run the data unit calculation sequence (DUCS), consolidate, and check the status on each Entity member. Therefore, including profit center in the Entity dimension will often be slower than using a User Defined dimension with typical data volumes.

Regardless of the approach, with the default, eliminations always occur at the first common parent in the Entity dimension. For a different approach, consider a non-matrix solution, which has separate cubes for statutory and management.

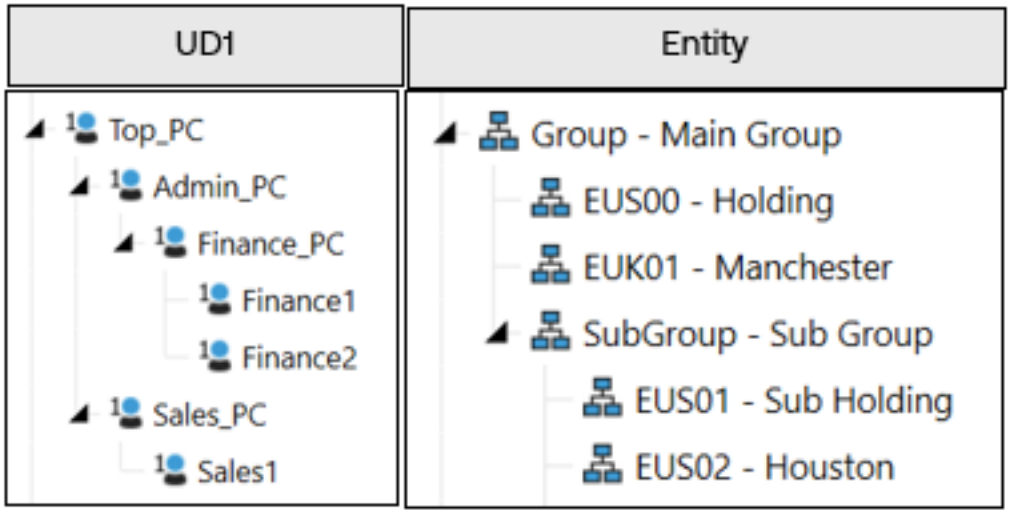
Since the Entity dimension option mostly uses system-default logic for processing and eliminating the data, the setup is straightforward. Therefore, the following information focuses on how to design for the User Defined dimension option to contain this detail and run eliminations.

Out-of-the-box View of Eliminations

If you need eliminations to run by profit center, it does not mean that you need to implement a full matrix consolidation solution. If you do not need the profit center elimination to happen at the first common parent in the PC hierarchy, out-of-the-box eliminations will suffice because you can report the eliminated data by selecting the correct combination of members, such as origin or PC.

Here is an example of the default behavior of eliminations in OneStream.

This is a profit center dimension in UD1 and an Entity dimension for the legal Entity members. In this example, all entities are using United States dollar (USD) only and are 100% owned.



There is an intercompany transaction between the legal entities Manchester and Houston. For Manchester, it is captured in the Finance PC. For Houston, it is captured in the Sales PC.

			Entity value	
			IC Payables	IC Receivables
Manchester	Houston	⊟ Finance1	500	
Houston	⊟ Manchester	Sales1		400

When out-of-the-box eliminations are run, the following results will display. Eliminations are in the red boxes, and consolidated results are in the blue box.

				Entity value		
				IC Payables	IC Receivables	IC Rec-Pay Plug
Main Group	Local	IC: Manchester	Profit Centre: Sales1		0	(400)
		IC: Houston	Profit Centre: Finance1	0		500
Manchester	Local	IC: Houston	Profit Centre: Finance1	500		
	Elimination	IC: Houston	Profit Centre: Finance1	(500)		500
Sub Group	Local	IC: Manchester	Profit Centre: Sales1		400	
		IC: Houston	Profit Centre: Finance1	0		
	Elimination	IC: Manchester	Profit Centre: Sales1		(400)	(400)
Houston	Local	IC: Manchester	Profit Centre: Sales1		400	

The eliminations occur at the first common parent in the Entity dimension. In this example, the first common parent is the Main Group. In UD1, the eliminations happen on the same member as the original data, so at the group level the plug amounts are by profit center.

Focus on the PC dimension (UD1) at the top Main Group reporting Entity member. The aggregated difference on the plug account of the base-level profit centers for Finance1 and Sales1 is 100.

				Entity value		
				IC Payables	IC Receivables	IC Rec-Pay Plug
Main Group	Local	IC: Top	Profit Centre: Top_PC	0	0	100
			Admin_PC	0		500
			Finance_PC	0		500
			Finance1	0		500
			Sales_PC		0	(400)
			Sales1		0	(400)

Matrix Consolidation - View of Eliminations

In this setup, you can apply matrix consolidation. With the same data, the counterparty PC for each transaction is captured.

				Entity value	
				IC Payables	IC Receivables
Entity: Manchester	IC: Houston	PC: Finance1	PC Counterparty: Sales1	500	
Entity: Houston	IC: Manchester	PC: Sales1	PC Counterparty: Finance1		400

In the following image, eliminations will happen on a new elimination member in UD1, rather than the member the data sits on, which is highlighted in the green boxes. For more information about the required elimination members, see UD1 – Profit Center.

The member where the elimination occurs represents the first common parent of the PC and Counterparty PC in the hierarchy. In this example, it is the Top_PC member in UD1.

				Entity value		
				IC Payables	IC Receivables	IC Rec-Pay Plug
Main Group	Local	IC: Manchester	Profit Centre: Sales1		400	
			Profit Centre: Elim_Top_PC		(400)	(400)
		IC: Houston	Profit Centre: Finance1	500		
			Profit Centre: Elim_Top_PC	(500)		500
Manchester	Local	IC: Houston	Profit Centre: Finance1	500		
	Elimination	IC: Houston	Profit Centre: Elim_Top_PC	(500)		500
Sub Group	Local	IC: Manchester	Profit Centre: Sales1		400	
	Elimination	IC: Manchester	Profit Centre: Elim_Top_PC		(400)	(400)
Houston	Local	IC: Manchester	Profit Centre: Sales1		400	

Review this result in more detail at the Main Group entity level. In the UD1 hierarchy, the elimination does not occur until the first common parent in the UD dimension. So, at Top_PC, the data is eliminated. But, at descendant UD1 members, the data is not eliminated, for example, Admin_PC, Finance_PC, and Sales_PC.

				Entity value		
				IC Payables	IC Receivables	IC Rec-Pay Plug
Main Group	⊟ Local	⊟ IC: Top	⊟ Profit Centre: Top_PC	0	0	100
			⊟ Admin_PC	500		
			⊟ Finance_PC	500		
			Finance1	500		
			⊟ Sales_PC		400	
			Sales1		400	
			Elim_Top_PC	(500)	(400)	100

The top PC and group entity have different approaches but the same result.

Setup

The following items are configured in the matrix consolidation example.

Entity

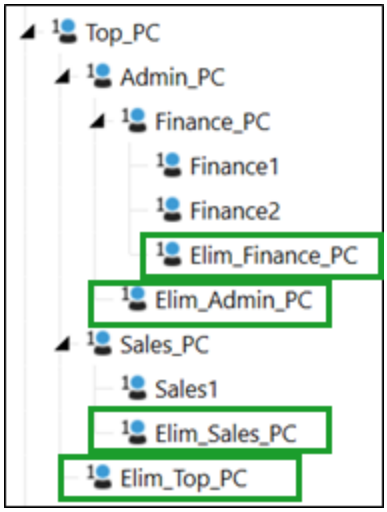
No changes are required to the Entity dimension for matrix consolidation.

Account

No changes are required to the Account dimension for matrix consolidation. The same plug accounts can be used.

UD1 – Profit Center

Some additional elimination members are required in UD1 . UD1 is used in this example. Standard design decisions apply to any User Defined dimension.



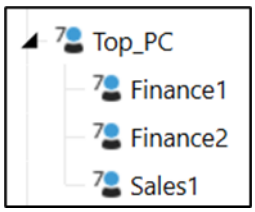
These new elimination members will be required at every point an elimination may happen, so you can see that this can add a large number of members to the existing hierarchy.

A common naming convention is often used to allow the system to derive where to post the elimination. In this case, it is the parent member name with the Elim_ prefix. Alternatively, you could use text fields to store this information. Either way, the logic will rely on this being updated accurately and consistently.

> **TIP:** Ensure your consolidation logic provides useful error messages if it finds that an elimination member does not exist or is misconfigured.

UD7 - New Counterparty Profit Center

A new dimension is needed to capture counterparty PC information. Like the Intercompany dimension in OneStream, this can be, and typically is, a flat list of the base counterparties.



All relevant intercompany data now needs to be analyzed by this dimension. Input forms and transformation rules will need to be updated.

In data models where almost all User Defined dimensions are already in use, this element can be challenging and requires consideration. UD7 is used in this example. The standard design decisions apply to any User Defined dimension. This dimension is used to capture the counterparty, so if your design already uses lots of dimensions, you may be able to combine this with other supplementary data or use UD8. However, this will require additional consideration in your dynamic reporting design.

> **TIP:** Consider how this dimension will be maintained. It is important for the logic that all members exist with the same naming in this counterparty dimension. Consider whether the counterparty dimension should be automated to align with the main profit center dimension.

Business Logic

Unlike the Entity dimension, all parents in a User Defined dimension are calculated on-the-fly. This approach requires additional eliminations to be calculated.

You will need to store your new matrix consolidation logic somewhere. In this case, it is a business rule attached to the cube, but it could also be attached to a Member Formula.

⊟ Business Rules	
Business Rule 1	InterUdElimination
Business Rule 2	

> **TIP:** You do not need to turn on a custom consolidation algorithm on the cube to achieve a matrix consolidation result. Consider the overall requirements and design.

Reporting

Custom reports need to be developed to enable intercompany matching and reporting on the resulting eliminations. If you already do eliminations like this, specifications should already exist that can be designed for this purpose. If not, you will need to understand matrix consolidation when you build reports and Quick Views or run legal entity-based reports with top for profit centers.

> **TIP:** You can use data set business rules to help efficiently gather data for interactive dashboards and reports.

Business Logic

Consolidation Algorithm

When a matrix consolidation requirement exists, it has been commonly observed that consultants will turn on the custom consolidation algorithm for each relevant cube. However, this stores the share data, so it has a negative impact on consolidation performance and database size.

Before you use the custom algorithm, consider calculating the matrix elimination adjustments during the calculation pass of C#Elimination within a UD member (potentially within your data audit dimension). This enables you to remain on the standard or org-by-period algorithm. Within this member, you can update the standard eliminations with the profit center detail.

You may have other requirements to use the custom algorithm. Determine the approach for matrix eliminations in the context of the overall design.

> **TIP:** Consider whether matrix eliminations are required for all processes and scenarios, and ensure it is only running on those that require it.

Useful Functions

The general approach for writing a matrix consolidation rule is to check that the elimination only occurs at the first common parent. Other than that, it follows standard OneStream rule writing techniques, such as using data buffers.

The following functions can be useful.

Function	Comment
api.Members.GetFirstCommonParent()	Use this function to check both the entity and PC parents to see if they are common to the IC or counterparty member.
api.Members.IsDescendant()	This does not check whether a descendant has a consolidation percentage greater than zero. So, if doing org-by-period, this may need additional consideration.
api.Entity.PercentConsolidation()	Use this function to check if entity is being consolidated. Ensure you only pass valid parent/entity combinations into the function.

Example Rule

The following rule is an example to demonstrate an approach to implement matrix consolidation. If you use it as a starting point, adapt and thoroughly test it before implementation.

The example takes the following approach:

- It retrieves a data buffer of the system-generated eliminations.
- It checks if the record is a new elimination.
- It reallocates the elimination (both intercompany and plug account entry) to the correct profit center.
- It clears the duplicate system elimination as No Data.
- It saves the results.

This should be assigned to the cube when using the standard or org-by-period consolidation algorithm. It reallocates the out-of-the-box eliminations to the relevant UD member.

With some reconfiguration of the dimensions and names referenced in the rule, it should work with the setup described.

```
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

Namespace OneStream.BusinessRule.Finance.MatrixElim
    Public Class MainClass
        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api
As FinanceRulesApi, ByVal args As FinanceRulesArgs) As Object
            Try
                Select Case api.FunctionType
                    Case Is = FinanceFunctionType.Calculate
                        If api.Pov.Cons.MemberId = DimConstants.Elimination Then

                            'Matrix consolidation
                            'Moves system-generated eliminations to first common parent elim
members In UD
                            'Moving eliminations will prevent original value and elimination
from netting, causing duplicate eliminations at each level
```

```
                            'Rule also clears these duplicate eliminations
                            'UD1 = main hierarchy (ProfitCentreDetail). UD7 = partner
hierarchy (PC_Interco)

                            'Get system-generated eliminations
                            Dim sourceDb As DataBuffer = api.Data.GetDataBufferUsingFormula
("RemoveZeros(A#All)",,False)
                            If sourceDb IsNot Nothing AndAlso sourceDb.DataBufferCells.Count
> 0 Then
                                Dim currentElimDb As New Databuffer

                                'Determine if consolidation algorithm is org-by-period
                                Dim povConsAlgorithm As ConsolidationAlgorithmType =
api.Cubes.GetCubeInfo().Cube.ConsAlgorithmType
                                Dim isOBP As Boolean = False
                                If povConsAlgorithm =
ConsolidationAlgorithmType.OrgByPeriodElimination OrElse povConsAlgorithm =
ConsolidationAlgorithmType.StoredShareAndOBPElimination
                                    isOBP = True
                                End If

                                For Each sourceCell As DataBufferCell In
sourceDb.DataBufferCells.Values
                                    '(1) Move system-generated eliminations to first common
parent elim members In UD

                                    '(1a) Record is new elim only if IC entity is not a
descendant of current entity
                                    Dim icPartner As String = sourceCell.GetICName(api)
                                    Dim isICDescendant As Boolean

                                    'Create dictionary to hold results of IsDescendant
check. Prevents repeating the test whenever the same IC partner is encountered
                                    Dim descDictionary As New Dictionary(Of String, Boolean)

                                    'Check if IC partner has previously been encountered. If
so, use previous IsDescendant results. If not, perform new IsDescendant check.
                                    If Not descDictionary.TryGetValue(icPartner,
isICDescendant)
                                        If Not isOBP Then
                                            'Standard algorithm
                                            isICDescendant = api.Members.IsDescendant
(api.Pov.EntityDim.DimPk, api.Pov.Entity.MemberId, sourceCell.DataBufferCellPk.ICid)
                                        Else
                                            'Org-by-period algorithm
                                            isICDescendant = IsDescendantWithNonZero(si,
api, api.Pov.EntityDim.DimPk, api.Pov.Entity.MemberId, sourceCell.DataBufferCellPk.ICid)
                                        End If
                                        'Save results of new IsDescendant check
                                        descDictionary.Add(icPartner, isICDescendant)
                                    End If

                                    '(1b) If record is new elim, move to first common parent
elim member
                                    If Not isICDescendant Then

                                        'Get first common parent of current record
```

```
                                        Dim ud1Pk As DimPk = api.Dimensions.GetDim
("ProfitCentreDetail").DimPk
                                        Dim topMemberId As Integer = api.Members.GetMemberId
(dimTypeId.UD1, "Top_PC")
                                        Dim partnerId As Integer = api.Members.GetMemberId
(dimTypeId.UD1, sourceCell.GetUD7Name(api))
                                        Dim commonParent As Member =
api.Members.GetFirstCommonParent(ud1Pk, topMemberId, sourceCell.DataBufferCellPk.UD1Id,
partnerId)
                                        If commonParent.Name.Length = 0 Then
                                            Throw New System.Exception($"No common parent
found for member {sourceCell.DataBufferCellPk.GetUD1Name(api)} and partner
{sourceCell.DataBufferCellPk.GetUD7Name(api)}")
                                        End If

                                        'Copy record and update UD to the UD elimination
member
                                        Dim targetCell As New DataBufferCell(sourceCell)
                                        Dim elimMember As Integer = api.Members.GetMemberId
(DimTypeId.UD1, $"Elim_{commonParent.Name}")
                                        If elimMember = -1 Then
                                            Throw New System.Exception($"Elimination member
for common parent {commonParent.Name} not found.")
                                        End If
                                        targetcell.SetUD1(api,$"Elim_{commonParent.Name}")
                                        currentElimDb.SetCell(si, targetCell, True)
                                    End If

                                    '(2) Clear duplicate eliminations
                                    Dim nodataCell As New DataBufferCell(sourceCell)
                                    nodataCell.SetData(0,datacellexistencetype.NoData,
nodataCell.CellStatus.StorageType)
                                    currentElimDb.SetCell(si, nodataCell, False)

                                Next

                                Dim di As ExpressionDestinationInfo =
api.Data.GetExpressionDestinationInfo("")
                                api.Data.SetDataBuffer(currentElimDb,di)
                            End If
                        End If
                End Select

                Return Nothing
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Checks if a given entity is a descenant with a non-zero consolidation %
        Public Function IsDescendantWithNonZero(ByVal si As SessionInfo, ByVal api As
FinanceRulesApi, ByVal dimPK As DimPk, ByVal ancestorMemberID As Integer, ByVal memberID As
Integer) As Boolean
            Try
                Dim isDescendant As Boolean = False
                If ancestorMemberID = memberID Then Return False
```

```
                Dim children As List(Of Member) = api.Members.GetChildren(dimPK,
ancestorMemberID)
                If (Not children Is Nothing) And children.Count>0 Then
                    For Each childMember In children
                        If api.Entity.PercentConsolidation(childMember.MemberId,
ancestorMemberID) <> Decimal.Zero
                            If childMember.MemberId = memberID
                                isDescendant = True
                                Exit For
                            End If
                            isDescendant = IsDescendantWithNonZero(si, api, dimPk,
childMember.MemberId, memberID)
                            If isDescendant Then
                                Exit For
                            End If
                        End If
                    Next
                End If

                Return isDescendant

            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

    End Class
End Namespace
```

Org-by-period in the User Defined Dimension

If you change the profit center structure, the Entity dimension has built-in tools to handle org-by-period so that entities can have relationship properties that vary by time period. Data is also stored on parent entities, which helps in this org-by-period. The User Defined dimension does not have the same functionality. So, if you move a member, that member is moved for all history. If you duplicate a member, the values are duplicated. This depends on the aggregation weight, but that cannot be varied by time.

If a profit center needs to change parents one month, consider these options:

- **Change the Main Hierarchy**: The old view will no longer be visible. The added complexity is that the eliminations for prior periods will occur in the wrong place in the UD hierarchy from a historical point of view unless a consolidation is rerun on all of those periods. If you rerun the consolidation on prior periods, all of your results will change, although not at the top level as long as nothing else has changed. This implies that the elimination will correctly display the elimination after the change. Historical data will not be kept for the reconsolidated periods.
- **Create Alternate Hierarchies (for example, Top_2023 and Top_2024)**: Create new hierarchies with unique parents to preserve the old hierarchy. As with the previous option, reconsolidation of prior periods is required to view historical data in the same format. However, if the data is only required in the new format for the future, reconsolidation of prior periods can be avoided.

> **TIP:** For each alternate hierarchy in which you run matrix eliminations, the eliminations will be duplicated. Therefore, your business logic should allow you to configure, by time period and scenario, which hierarchies are eliminated to ensure only the necessary calculations are run. This could be done, for example, through tags on text fields of the members.

It is not a common scenario, but it is something to consider during the requirements gathering and design.

Equity Pickup

OneStream supports equity pickup calculations using three different properties located under the entity's settings, all of which can vary by scenario type. These settings were designed to implement equity pickup using normal business rules and formulas.

See *Entity Dimension* in Cubes for definitions of each setting.

Equity Pickup Example

The entity hierarchy example below illustrates how equity pickup works in OneStream and discusses how each entity's currency and equity pickup settings affect calculation and consolidation.

- Clubs (USD)
- Holding (EUR)
- Houston (USD)
- Carlsbad (USD)
- Frankfurt (EUR)

In the example above, the formulas for Holding need to read calculated data from the other sibling entities. Therefore, the Sibling Consolidation Pass property for Holding would be set to Pass 2, causing calculation to occur on all the other sibling entities before the Holding entity is calculated. This allows the formulas for Holding to correctly read calculated data from Houston, Carlsbad, and Frankfurt. For entities not involved in equity pickup, the (Use Default), or Pass 1 settings for Sibling Consolidation Pass causes all sibling entities to be calculated at the same time.

Holding is using a different local currency than its parent clubs, but only wants to read data using the EUR currency. In this situation, the Auto Translation Currencies setting for Houston and Carlsbad needs to be set to EUR in order to have them automatically translate to EUR when Clubs is consolidated. Normally, all the sibling entities translate to the parent entity's local currency, which in this case is USD, however this setting tells the engine to translate Houston and Carlsbad to EUR as well during the consolidation. Once the consolidation is complete, Holding's formulas, which are calculated in Pass 2, can read data from E#Houston:C#EUR, E#Carlsbad:C#EUR, and E#Frankfurt:C#EUR.

Sibling Repeat Calculation Pass is designed for circular ownership and may not be used as often as the other equity pickup settings. If this is used, it causes the entity's calculation to be repeated after all the Sibling Calculation Passes have been completed. For example, if there was another entity in the structure above named Holding2, it would be set to use a Sibling Calculation Pass of Pass 3. This would cause its normal calculation to occur after Holding and allow Holding2 to read calculated data from Holding. Holding could also use a Sibling Repeat Calculation Pass causing it to be recalculated. In that repeat calculation, Holding could then read calculated data from Holding2 resulting in circular ownership. When writing formulas, use api.Args.CalculateArgs.HasRepeatCalc and api.Args.CalculateArgs.IsRepeatCalc to determine if the engine is currently running the repeat calculation.

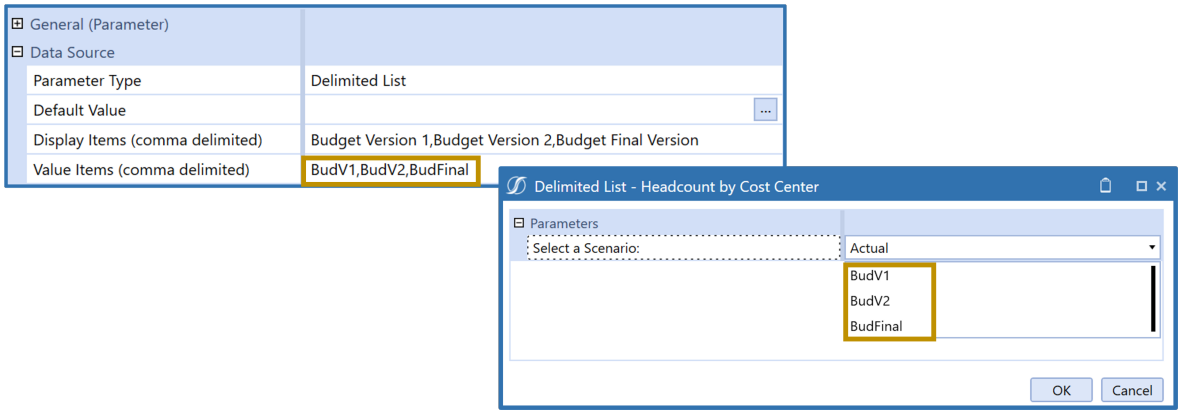
Filter Runtime Data and Reduce Maintenance with Parameters

Parameters prompt users to filter or customize data at runtime, helping streamline objects such as forms, dashboards, and Cube View reports to best suit different user needs. Administrators typically define parameters primarily in maintenance units, and then assign parameters to the appropriate object. See:

- About Parameters
- Parameter Types
- Benefits
- Ways to Use Parameters
- Requirements
- Best Practices
- Create Parameters

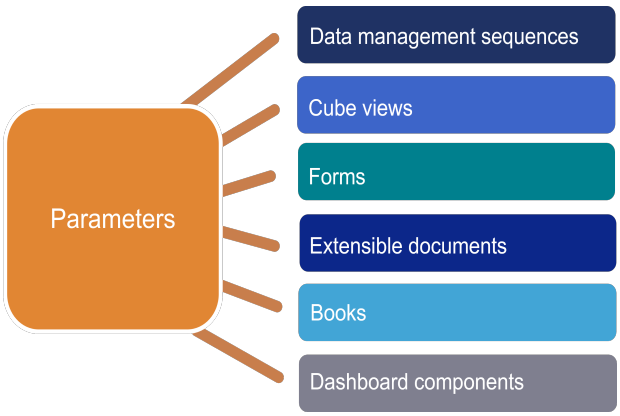
About Parameters

Depending on the type, parameters prompt you to specify data by entering text or selecting items from a drop-down menu or dialog box. For example, a Delimited List parameter could prompt you to select from 1 of 3 budget versions to use in a Headcount by Cost Center report, as shown:



As shown below, you can use parameters with a variety of objects, such as:

- Data management sequences
- Cube Views
- Forms
- Extensible Documents
- Books
- Dashboard components



For example, use parameters as placeholders or with substitution variables and retrieve functions to enhance Extensible Documents with selections for time period, entity, data cell values, and more. See Ways to Use Parameters.

Parameter Types

Parameters are created in dashboard maintenance units. The following parameter types are available, each with specific uses.

- Literal Value Parameters
- Input Value Parameters
- Delimited List Parameters
- Bound List Parameters
- Member List Parameters
- Member Dialog Parameters

To reference an existing parameter, when formatting a Cube View header or cell for example, enclose the parameter name in pipes and exclamation points. For example, |!ParameterName!|.

Each parameter type has two sections of properties, General and Data Source parameter properties. Properties identify a parameter, determine a parameter's behavior, and control how the parameter is used.

General properties are standard across all parameter types. Data Source properties vary by parameter type, but the first property for all parameter types is the Parameter Type property, where you select the type of parameter from a list.

Literal Value Parameters

Literal Value parameters are the only parameter type that does not prompt for any selections during creation.

Use Literal Value parameters to standardize formatting for Cube View columns and rows and dashboard components. This helps reduce reporting maintenance.

To create a literal value parameter, first create a parameter, then specify the Literal Value parameter settings. See Create a Parameter and then Specify Literal Value Settings.

Input Value Parameters

Use Input Value parameters to flexibly enter or change a value used in a Cube View or other object. You can enter or modify a value to use in dashboard components and Cube View rows and columns.

The Input Value parameter type helps you standardize and more easily maintain report and dashboard formats. For example, you can create an Input Value parameter that, when run in a Cube View, prompts a for the name to display as Report Run By in a product sales report.



	Budget V1	Budget V2	Final Budget
Total Products	534,686.10	534,686.10	534,686.10
GolfBalls	360,843.60	360,843.60	360,843.60
Tour	109,858.60	109,858.60	109,858.60
Premium	81,400.00	81,400.00	81,400.00
Range	45,570.00	45,570.00	45,570.00
Logo	83,995.00	83,995.00	83,995.00
LogoTour	40,020.00	40,020.00	40,020.00
Clubs	143,019.50	143,019.50	143,019.50
Woods	106,931.00	106,931.00	106,931.00
Drivers	50,400.00	50,400.00	50,400.00
FairwayWoods	56,531.00	56,531.00	56,531.00

To create an Input Value parameter, first create a parameter, then specify the Literal Value parameter settings. See Create a Parameter and then Specify Literal Value Settings.

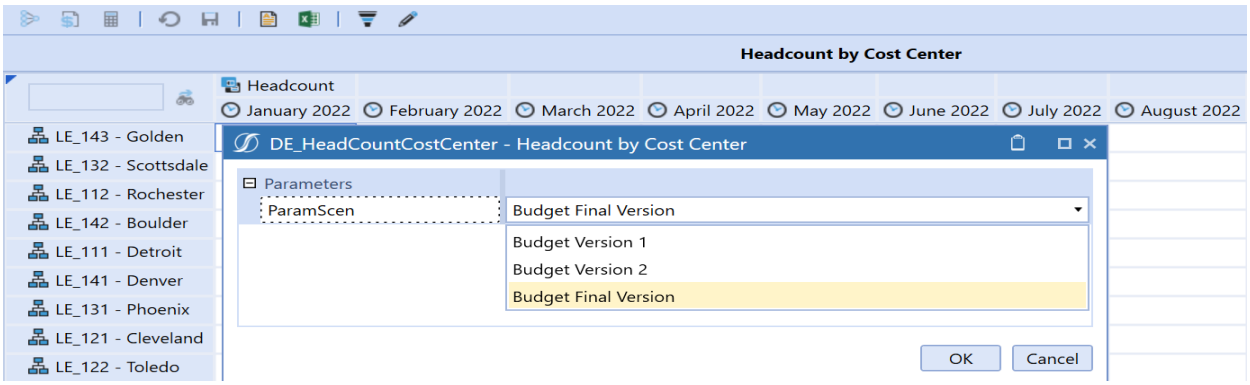
Delimited List Parameters

Use Delimited List parameters to create a distinct list of specified values, including an easily recognizable name. For example, you can create a list containing the months of the year to use as value items. Use the display items field for the Delimited List parameter to list each month value separated by a comma, then associate each with a value item to create the list.

Delimited List parameters let you pick a value from a drop-down menu instead of typing a value into a cell. Parameters can be assigned as a list source to a Cube View row or column. These parameters are supported in Excel, Modern Browser Experience, and Cube View-driven reports.

You can also specify a parameter name that can be a Cube View row or column. Edit cells using a drop-down menucontaining the parameter's list of items. A number is stored in the data cell as specified in the parameter's definition. If using a Delimited List parameter on a numeric cell, ensure each value in the parameter's name-value pairs is a number.

Select from a list of values, specified as display or value items, such as months or financial quarters. For example, a Delimited List parameter could prompt a user to select from one of three budget versions for a Headcount by Cost Center report.



These types of parameters also help you prompt for columns when sharing columns in a Cube View, so you can specify column names at runtime. For example, you could specify columns for products or years in an income statement.

To create a Delimited List parameter, first create a parameter, then specify the Literal Value parameter settings. See Create a Parameter and then Specify Literal Value Settings.

> **Example:** See Sample Delimited List Parameter in a Cube View.

Bound List Parameters

A bound list is group of members or other objects that were created using a predefined method query or by entering an SQL expression to retrieve members. Use Default Value to enter an input value.

For example, to list all entities in a dimension, use this method query with a command type of member:

```
{Dimension type}{Dimension name}{Member Filter}{where clause}
```

```
{Entity}{CorpEntities}{E#Root.TreeDescendants}{}
```

The difference between the two query types is that a method query is an SQL query helper that you can use to quickly perform queries.

SQL queries are more customizable and often used in dashboard design to prompt for information such as the Workflow Profile name. See "Determine the Query Method" in the *Data Collection Guide*.

To create a Bound List parameter, first create a new parameter, then specify the Bound List parameter settings. See Specify Bound List Settings.

Member List Parameters

A Member List parameter is similar to a Delimited List parameter, but you must specify more properties. Also, a Member List parameter does not have the option to display a value item or display item.

Users select a member, such as an account or entity dimension member. For example, you could prompt users to select a base Scenario member when they run a Cube View to display budget actuals.

Members that display in the member list are based on the member filters you specify for a dimension.

To create a Member List parameter, first create a new parameter, then specify the Member List parameter settings. See Specify Member List Settings.

Member Dialog Parameters

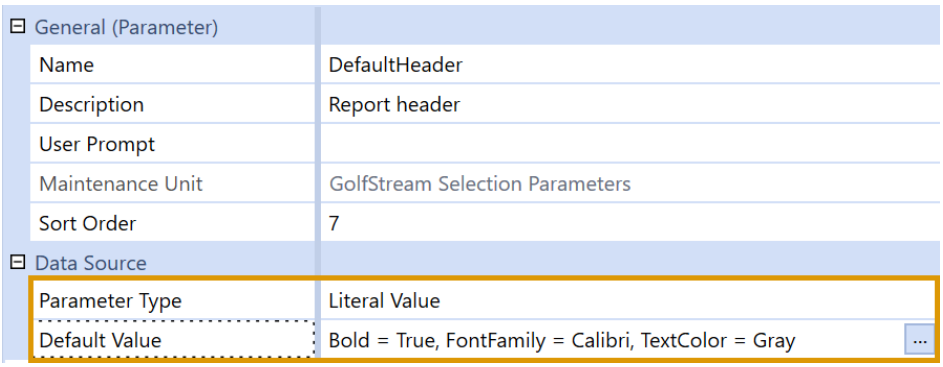
Similar to Member List parameter, the Member Dialog parameter lets you associate a member with the parameter through a dialog box with search capabilities. This is more appropriate for a dimension such as accounts or entities where you can use the hierarchy to select a base or parent member.

To create a Member Dialog parameter, first create a new parameter, then specify the Member Dialog parameter settings. See Specify Member Dialog Settings.

Benefits

As configurable prompts, parameters enhance reporting and analysis flexibility because they let you dynamically filter the data to retrieve. You can also use some types of parameters with form templates to streamline data collection. Some parameters also store values that you can share across objects, helping make applications more dynamic and reduce some maintenance tasks. For example:

- Use parameters with substitution variables and retrieve functions to hold values so you can insert time periods, entities, text comments, and data cells in Extensible Documents.
- Use Literal Value parameters to define and apply standard formatting to Cube View rows and columns and to dashboard components, reducing repetitive, error-prone tasks and ensuring a consistent look and feel in your analytics. You could create a Default Header parameter that captures the header format to use in reports as shown below and then assign the parameter to the appropriate Cube Views.



For information about some of the tasks you can perform with parameters, see Ways to Use Parameters.

Ways to Use Parameters

You can use parameters extensively across OneStream objects. This topic identifies some of the advantages of using parameters with some objects such as the following with:

- Cube Views
- Data Adapters and Dashboard Components
- Extensible Documents
- Books

Cube Views

Assign parameters to a Cube View to:

- Specify and apply standard formatting to Cube Views and dashboards for a consistent look and feel in your reports.
- Identify the details users can access when they navigate linked Cube Views and dashboards to access more data.See "Linked Cube Views" in the *Presentation Guide*.
- Let users refine the data to supply or retrieve in reports by prompting them to select entities, Workflow Profiles, views, scenarios, and more. To limit a query, refer to the parameter in the point of view, rows, or columns. This filters data in reports based on parameter selections.
- Specify the Point of View to display more focused data.
- Nest parameters to use a series of parameter prompts to refine the data to use on dashboards and in reports. See Nest Parameters.

Data Adapters and Dashboard Components

Use parameters with data adapters to:

- Prompt users to select the data to retrieve and display on dashboards, so they can focus their analysis using custom intersections of some of the Cube View data that populates the dashboard. For example, assign parameters to prompt for entity and year selections to a data adapter so users can visualize data by entity and year while accessing the rest of the Cube View data.
- Enable dashboard actions such as custom calculations or launching another dashboard.
- Identify static values using Literal Value parameters, for dashboard content and design.

See "Dashboard Components" in the *Presentation Guide*.

Extensible Documents

Use parameters as placeholders and with substitution variables and retrieve functions to enhance extensible documents, such as Word and PowerPoint files, with dynamic selections for time period, entity, data cell value, text comments, and more. For example, insert the following parameters to prompt users to specify the years and quarter to use in a report:

- |!CurrentQtr!|
- |!ReportingYear!|
- |!ReportingYearPrior!|

You can use parameters in Microsoft text, Excel, Word, and PowerPoint files. See "Present Data with Extensible Documents" in the *Presentation Guide*.

Books

Use books to combine a variety of reports and files in one object for custom reporting. You can automatically generate input parameters based on book content or manually specify the required parameters. You can also use parameters in loops- sequences of variable-based instructions, to further customize the data to display. For example, a book can loop to display all base entities in a hierarchy, providing the same Cube View report for each entity. Or, you could assign a parameter called *ParamSalesRegions* to include all sales regions in book content.

If the source objects used in a book have assigned parameters, use Change parameters to override them instead of modifying the underlying Cube View or report. For example, configure a Change parameter to use the workflow time in a book instead of prompting users to specify a Time dimension member.

> **NOTE:** Before creating books and using Change parameters, ensure that you understand the source objects, such as Cube Views, that generate the content to use.

Requirements

This topic describes who should create parameters and the syntax to use when you assign parameters to application objects.

Definition and Assignment

Because they are more familiar with OneStream applications, objects and processes, a OneStream Administrator or an application designer should define parameters after identifying:

- The information that a parameter must gather from users at runtime.
- If a sequence of parameters is needed to gather the required data. Specify a staggered sort order to display parameters in the right order to collect increasingly granular levels of detail.

See Best Practices.

Syntax

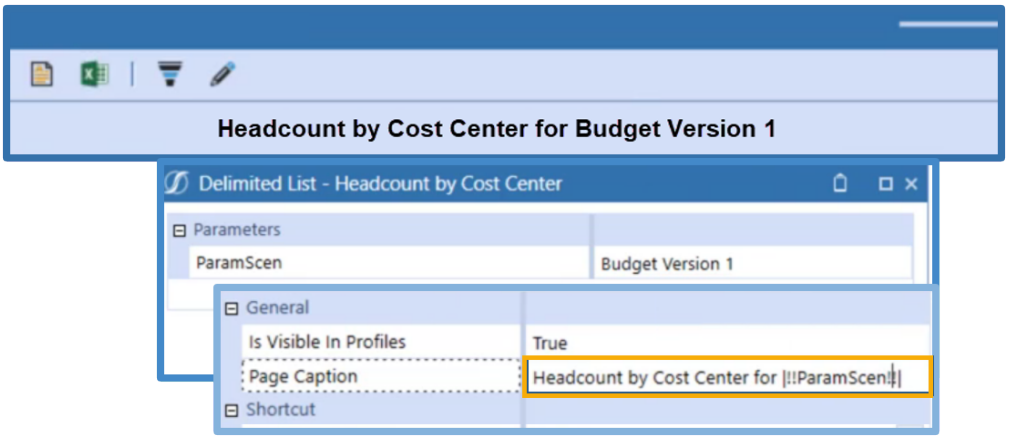
Enclose parameter names in pipes and exclamation points when you assign parameters to objects, regardless of parameter type. For example, |!ParamView!| prompts users to select year to date or periodic data when they run a Cube View for reporting.

Use double exclamation points (for example, |!!ParamEntity!!|) when assigning Delimited List parameters to present users with the parameter's more intuitive and helpful display item value.

Follow these best practices when assigning Delimited List parameters:

- Assignment to Cube View rows and columns - Use single exclamation points to display the member name. If you use double exclamation points, you may receive an error at runtime as the parameter tries to retrieve the Display Item value.

- Assignment to page captions - Use single exclamation points to display the Value Items. Use double exclamation points to show the Display Item. For example, instead of displaying BudV1 - the Value Item, specify |!!Budget Version 1!!| to display Budget Version 1 on reports as shown:



Best Practices

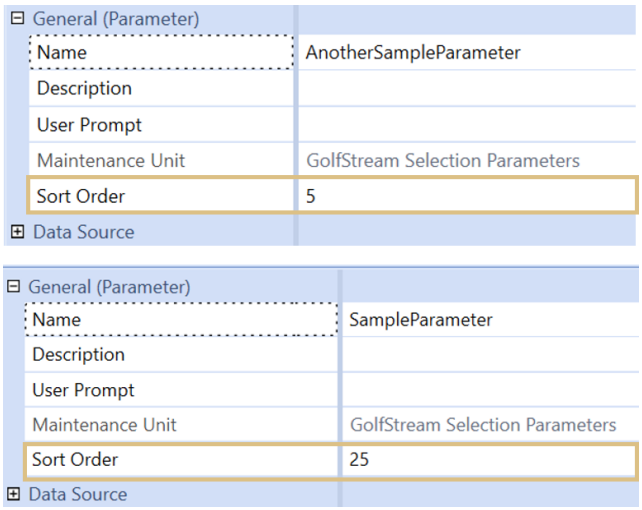
This topic provides tips to help you design and maintain parameters. See:

- Design
- Naming

Design

Since parameters can be used with different objects such as similar types of reports, reduce the associated maintenance tasks by designing parameters with:

- Reuse in mind, making parameters applicable to different objects. If you modify a parameter to address changing reporting needs, keep in mind that your changes apply to all objects to which the parameter is assigned.
- A staggered sort order where at least 20 sort numbers are unassigned to support new parameters possibly needed for future report and dashboard redesign.
  For example, two parameters with respective sort orders of five and six display consecutively. However, to display a new parameter in between them, you must change the parameter with sort order of 6 to 7 and then assign sort order of 6 to the new parameter. If you have hundreds or parameters, changing the existing sort order to support new parameters is tedious and error prone. In this example, assign a sort order of 5 to the first parameter and 25 to the second parameter to easily add new parameters in between them.



Naming

- To quickly identify parameters, prefix parameter names with Param. For example, ParamProductSegment, ParamCostCenter, and ParamColumnColor.
- Do not use special characters or dashes (-) in parameter names. Also, specify intuitive names for parameters. If you rename a parameter, you must re-assign it to all, potentially hundreds, of its associated objects.
- Parameter names should be unique, even across Workspaces.

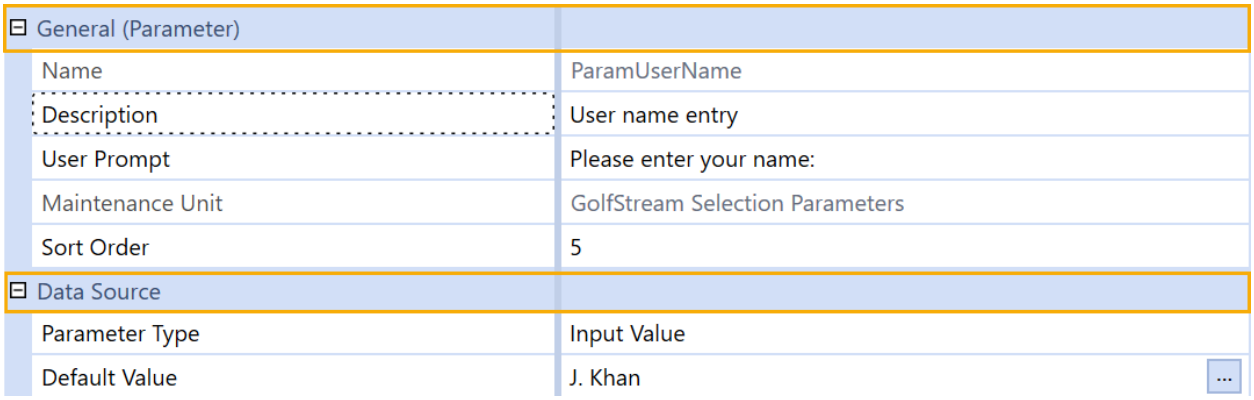
About Parameter Definitions

This topic orients you with the properties you specify to create parameters and how to look up objects and values to specify as the default value of a parameter.

Property Types

You specify two types of properties to create a parameter, as shown below:

- General properties, such as user prompt and sort order, that are common to all parameter types.
- Data source properties that are parameter type-specific. These properties define the value or objects to apply with the parameter or the prompt options that users can select at runtime.



See Create a Parameter.

Create a Parameter

Parameters prompt users to filter or customize data at runtime, helping tailor objects such as forms, dashboards, and Cube Views to best suit different user needs. Administrators define parameters in dashboard maintenance units, then assign them to the appropriate objects.

> **TIP:** To further refine data selections, refer one parameter to another. See Nest Parameters.

1. Review the Requirements and Best Practices.
2. Select **Application** > **Presentation** > **Workspaces**

   > **TIP:** You can also define parameters by selecting **System** > **Administration** > **Workspaces**.

3. Select the dashboard maintenance unit in which to create a parameter, then **Parameters**. After you save a parameter, you cannot put it in another dashboard maintenance unit.

4. Select the  **Create Parameter** icon in the toolbar.

5. In the **Name** and **Description** fields, enter a name and description that indicates the information the parameter prompts users to specify, such as ParamBudget Scenario. Do not use special characters, dashes (-) or underscores (_) in names. Names should be unique, even across Workspaces.

   See "Naming" in Best Practices for tips on naming and organizing parameters.

6. In **User Prompt**, enter the message that prompts users to specify information. For example, Select a Product member or Enter your name.

   Skip this step for literal value parameters since they do not prompt for selections.

7. In **Sort Order**, enter the sequence in which to display the parameter. Parameters are displayed by alphabetical order by sort order and name.

   - In the list of parameters for the dashboard maintenance unit. For example, to list the parameter sixth, enter six.
   - For use with other parameters. For example, to display parameterB after parameterA, ensure that the sort order value for parameterB is greater than that of parameterA. See Best Practices.

8. From **Parameter Type**, select the kind of parameter to define which determines the information or objects users are prompted to specify.See Parameter Types.

   - **Literal Value**: Instead of prompting users for a selection, use a predefined value or object such as a color or number format.
   - **Input Value**: Users enter or modify a value to customize Cube View results that generate reports. For example, prompt users to enter their name when they run a Cube View to generate a report.

- **Delimited List**: Users select one value or item from a list, such as one of two budget scenarios.
- **Bound List**: Users select a member from a list of members generated with a method or a SQL query.
- **Member List**: Users select a member from a drop-down menu.
- **Member Dialog**: Users select a member by navigating, searching, and filtering the member hierarchy displayed in a dialog box.

9. Define the settings for the parameter type:
    - Specify Literal Value Settings.
    - Specify Input Value Settings.
    - Specify Delimited List Settings.
    - Specify Bound List Settings.
    - Specify Member Dialog Settings.
    - Specify Member List Settings.
10. Select the **Save** icon.
11. Add the parameter to the appropriate object, such as a Cube View. See Assign a Parameter.

Specify Literal Value Settings

To finish defining a Literal Value parameter:

1. In **Default Value**, specify the value or object to apply with the parameter. Users cannot change this at run time. Enter a value (color or currency, for example) or an object name, or

select **Edit** to open object lookup.

2. Select the **Save** icon.

> **IMPORTANT:** When you assign the parameter, enclose the name in pipes and exclamation points. For example, |!ParamColumnColor!|.

Specify Input Value Parameter Settings

To finish defining an Input Value parameter:

1. (Optional) Although users can override it at runtime, use **Default Value** to specify a value or object to initially display the first time the parameter runs. Enter the object name or value, or select **Edit** to use object lookup.
2. Select the **Save** icon.

> **IMPORTANT:** When you assign the parameter, enclose the name in pipes and exclamation points. For example, |!DefaultHeader!|.

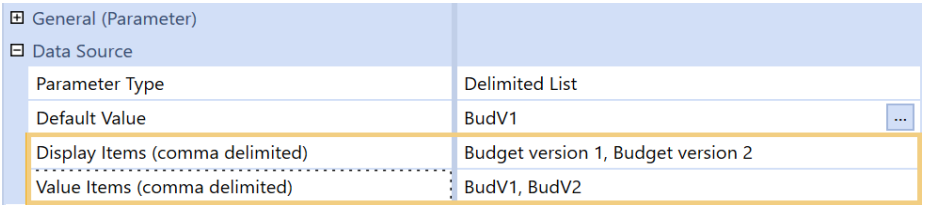
Specify Delimited List Settings

To finish defining a Delimited List parameter:

1. In **Default Value**, specify a value or object to initially display the first time the parameter runs. Enter the member name or select **Edit** to use object lookup.
2. In **Display Items** (comma-delimited), define a custom, comma-delimited list to display several options in the application from which to select, for example Budget Version 1, Budget Version 2. Separate names with commas.

3. In **Value Items** (comma-delimited), enter the values or objects, such as member names, that correspond to each Display Item option in the application. Enter options in a comma-delimited list using their names in the application. Separate the values with commas.

   For example, if the Scenario dimension member for Budget Version 1 is BudV1, enter BudV1. Similarly, enter BudV2 as the member used if Budget Version 2 is selected. Separate names with commas as shown:



The syntax you use when you assign the parameter to an object depends on:

- The object type and component, such as a Cube View's page caption or row.
- If you want to present the Value Item or Display Item to users at run time.

See Requirements.

> **NOTE:** If a Delimited List parameter is used in a Cube View page caption, surround it in two exclamation points to reference the display items, not the value. For example, |!!ParameterName!!|.

4. Select the **Save** icon to save your property edits.

Specify Bound List Settings

You can define two types of Bound List parameters:

- An SQL-based query to leverage data in the OneStream application or framework database or in an external data source.
- A method-based query to use object-specific variables to refine and customize the member data to return.

To finish defining a method-based parameter:

1. In **Default Value**, enter the actual value used by default in the parameter, if no other value is specified. The default value must be part of the parameter's definition.

   You can either type the default value if known, use object lookup to create the default value, or use a Cube View to select the appropriate formatting and copy the result to the Default Value field.

   The default value property is the member you would set before defining a value using the data source properties specific to the selected parameter type.

2. Use **Result Format String Type** to determine if the default value or a custom value is used.

   - **Default**: Use the formatting defined in the Default Value property.
   - **Custom**: Use a format string that you enter in Result Custom Format String as the display text for the parameter.

   Selecting **Custom** enables the **Result Custom Format String** property, where you can design the custom string.

3. In **Result Custom Format String**, optionally enter a string to use as the display text. Use `String.Format` syntax in `.NET`. For example, if a value is Blue and the format string is `The color is {0}.`, the display text is `The color is blue`.

4. In **Command Type**, choose one of the following:

- **Method** to use a specified method type and query to define the command to be used in the parameter.
- **SQL Query** to determine the database to use and define the SQL query.

Both selections enable additional properties to further define the command.

For a Method Command Type: Use these properties to define the database location if using a Method command type.

- **Method Type**: Select the object-specific variables to define how members are evaluated, so the most useful members are returned. See "Data Adapters" in the *Presentation Guide* for more information on using the method type in a parameter definition.
- **Method Query**: Customize variables to define which members to return and display in prompts. You can either type the default value if known, or use object lookup to create the default value. Use this syntax:

  ```
  {Dimension type}{Dimension name}{Member Filter}{where clause}
  ```

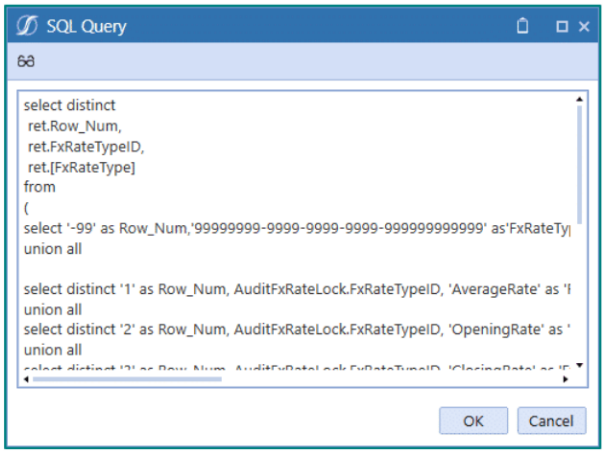
For a SQL Command Type: Use these properties to define the database location if using a SQL command type.

- **Database Location**: Determines the location of the database that includes the SQL command type. Select one of the following:
  - **Application**: Use the current OneStream Application database containing stage and financial cube data.
  - **Framework**: Use the connected OneStream Framework database containing security and log data.
  - **External**: Indicates that a database outside of OneStream is to be used.

If selecting **External** for the database, use the enabled **External Database Connection** property to select from the list of external databases. External database connections are defined in OneStream using the Server Configuration Tool. Select the external database connection you want to use from the list.

See "Configuring Application Server" in the *Installation Guide* for information on configured application servers.

- **SQL Query**: This is the SQL statement to run for the parameter. Enter a query that evaluates objects, such as members, to return a particular set. For example:



> **NOTE:** If you do not know the method query syntax, leave this field blank and select . This identifies the syntax location, runs the parameter, and provides a sample result set.

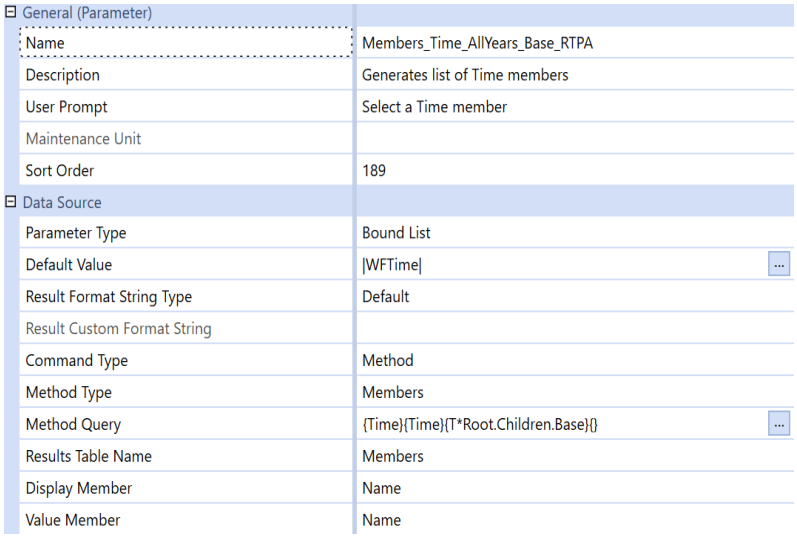
5. In **Results Table Name**, enter the name of the table generated when the data adapter runs. The default name is table.
6. In **Display Member**, enter one of the following:

- **Name** to display the member by name.
- **Description** to display the member's description.
- **Name and Description** to display both the member name and the member description.

7. In **Value Member**, enter the name of the members, as used in the application, to correspond to each member displayed.
8. Select the **Save** icon to save your property edits.

Sample Method-Based Parameter

The following image shows the definition of a parameter that retrieves Time members for an Application Analysis report.



General (Parameter)	
Name	Members_Time_AllYears_Base_RTPA
Description	Generates list of Time members
User Prompt	Select a Time member
Maintenance Unit	
Sort Order	189
Data Source	
Parameter Type	Bound List
Default Value	|WFTime|
Result Format String Type	Default
Result Custom Format String	
Command Type	Method
Method Type	Members
Method Query	{Time}{Time}{T*Root.Children.Base}{}
Results Table Name	Members
Display Member	Name
Value Member	Name

Sample SQL-Based Parameter

The following image shows the definition of a parameter that generates a list of FX Rate Types for an FX Rates Audit report.

⊟ General (Parameter)	
Name	FXRateTypes_RTPA
Description	Generates FX Rate types
User Prompt	
Maintenance Unit	
Sort Order	105
⊟ Data Source	
Parameter Type	Bound List
Default Value	(All) ...
Result Format String Type	Default
Result Custom Format String	
Command Type	SQL
Database Location	Application
SQL Query	union all ...
Results Table Name	FXRateTypes
Display Member	FXRateType
Value Member	FXRateTypeID

Specify Member List Settings

To finish defining a Member List parameter:

1. (Optional) In **Default Value**, specify a member to display at the top of the list when the parameter runs and prompts users to select a member. Enter the member name or select **Edit** open object lookup.
2. Right-click in the **Default Value** dialog box, select **Paste** and then select **OK**.
3. Use **Display Member** to define how to display the returned members.
    - Enter **Name** to display the member by name.
    - Enter **Description** to display the member's description.
    - Enter **Name and Description** to use both.
4. In **Cube**, either type the default value if known, or select **Edit** to use object lookup to select the cube containing the required dimensions.

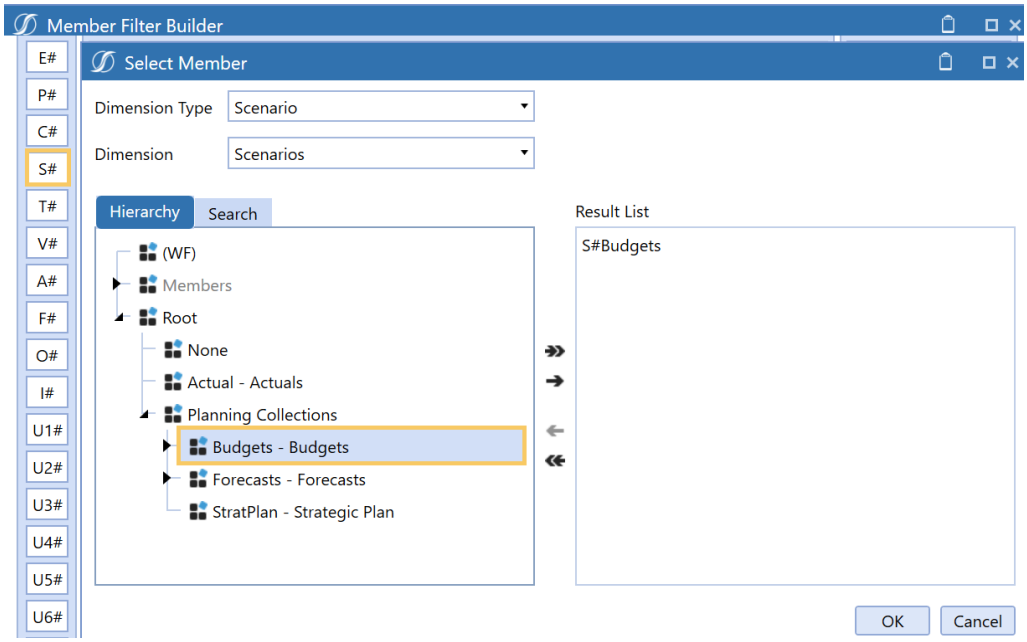
5. From **Dimension Type**, select the dimension category,such as Time or Account, that contains the appropriate dimension.
6. In **Dimension**, enter the name of the dimension to use or select **Edit** to select the dimension in the hierarchy.
7. In **Member Filter**, either type the Member Filter if known or select **Edit** to use the Member Filter Builder to specify the members to list. See "Member Filter Builder" in the *Presentation Guide* for more information.
8. Select the **Save** icon to save your changes.

Specify Member Dialog Settings

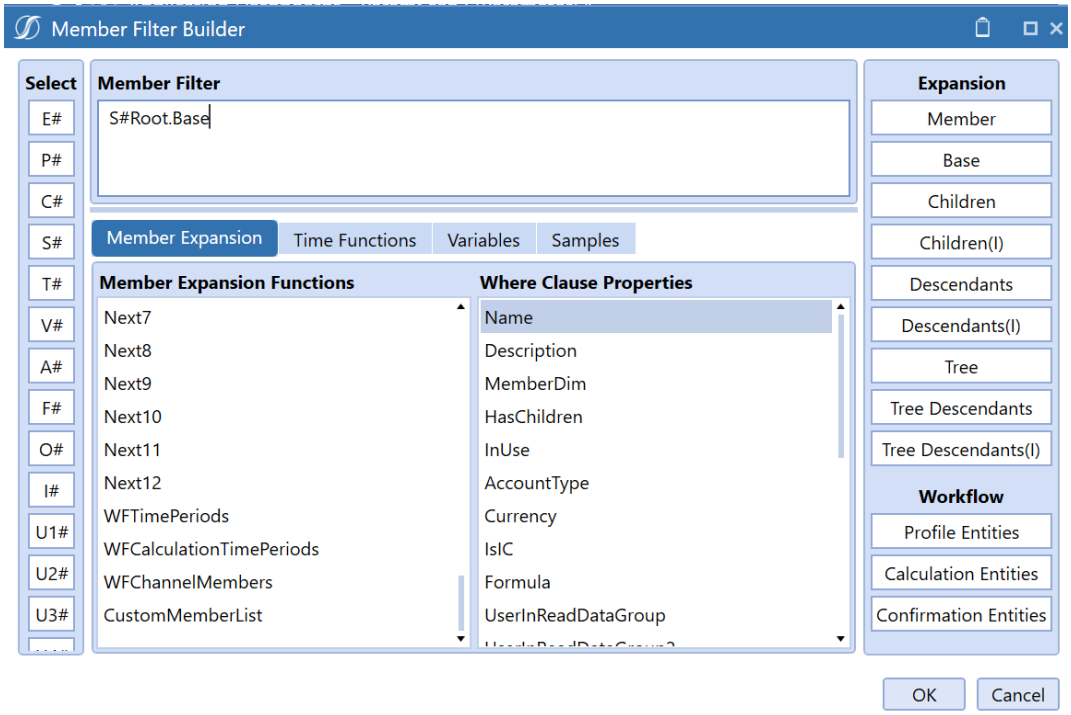
To finish defining a Member Dialog parameter:

1. (Optional) In **Default Value**, specify a dimension in the dimensional hierarchy to display when the parameter first runs. Enter the dimension name or select **Edit** to open the object lookup.
2. In **Cube**, select **Edit** to select the cube that uses the required dimension. To quickly find a cube, enter the name in **Filter**.
3. From **Dimension Type**, select the dimension category such as Consolidation or Scenario that contains the appropriate dimension.
4. In **Dimension**, enter the name of the dimension to use or select **Edit** to select the dimension from the hierarchy.
5. In **Member Filter**, specify the member that determines the hierarchy to display in parameter prompts. Enter a member name or select **Edit** to use the Member Filter Builder. See "Member Filter Builder" in the *Presentation Guide*. For example:

- To display only Budget members in a Scenario dimension with multiple member groups such as Actual and Forecast, select **S#** and select **Budgets** as shown:



- To display the entire hierarchy, select **Root** and **Base**.



> **NOTE:** Dashboard-based parameters are similar to those used in forms. If a form references a parameter that cannot be found in the form template, a dashboard parameter of the same name is used.

6. Select members and use the arrows to add them to **Results**.
7. Select **OK** twice.
8. Select **Save**.

> **IMPORTANT:** When you assign the parameter, enclose the name in pipes and exclamation points. For example, |!BudgetScenario!|.

Sample Delimited List Parameter in a Cube View

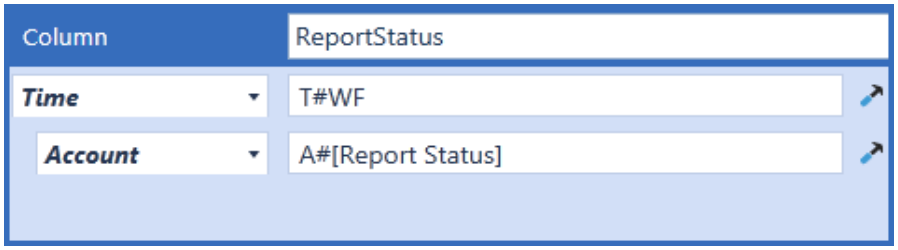
In this example, a Delimited List parameter is created and assigned to a sample GolfStream Cube View so users can indicate if reports are not started, under review, or completed for each entity.

1. Select **Application** > **Presentation** > **Workspaces** and then select the dashboard maintenance unit where you want to store the parameter.
2. Select **Create New Parameter**.
3. Define the parameter by specifying the settings shown below. Enter an intuitive name to reflect the parameter's purpose of prompting users to select from a list of report status settings.

⊟ General (Parameter)	
Name	ParamReportStatus
Description	Status selection parameter for report analysis
User Prompt	Report status:
Maintenance Unit	GolfStream Selection Parameters
Sort Order	82
⊟ Data Source	
Parameter Type	Delimited List
Default Value	...
Display Items (comma delimited)	Not Started, Under Review, Completed
Value Items (comma delimited)	0NotStarted, 1Review, 2Complete

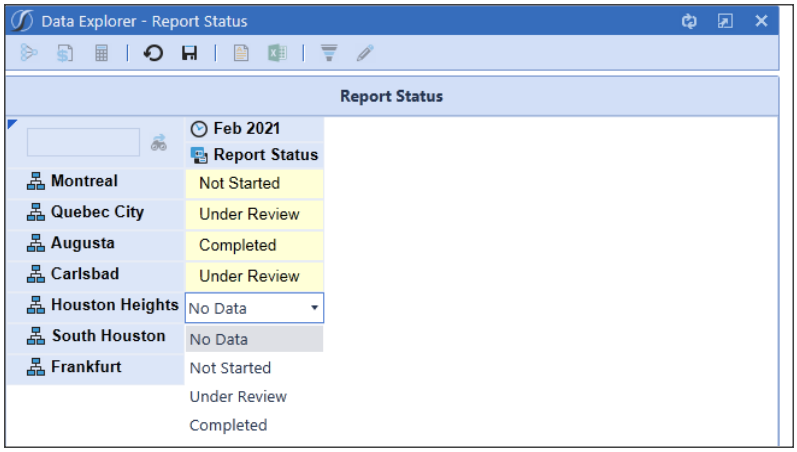
4. In **Display Items**, enter the status options that users can select when they run the Cube View.
5. In **Value Items**, enter the corresponding status member names defined in the application. For example, 0NotStarted is the Value Item for the Display Name Not Started.
6. Select the **Save** icon.

7. Select **Application** > **Presentation** > **Cube Views** and select the Cube View to use with the parameter.
8. Select **Rows and Columns** to see that column members display time based on workflow and a non-financial account called Report Status. The row members identify all base entities for Clubs.



9. Selecta row header and then **Data**.
10. In **Cell Type**, select **Combo Box**.
11. In **List Parameter**, enter **ParamReportStatus** and then save the Cube View.

    When users run theCcube View, they can specify a report status for each entity, as shown:



Manage Parameters

This topic describes the tasks administrators can perform to maintain parameters. See:

- Assign a Parameter
- Modify a Parameter
- Copy a Parameter
- Delete or remove a Parameter
- Rename a Parameter
- Identify Parameter Assignments
- Nest Parameters

Assign a Parameter

After defining a parameter, assign it to the object where it should be used to filter runtime data or apply a stored value.

1. Select **Application** and then the appropriate object.
2. Edit the object to assign the parameter. Select **Object Lookup** to find the parameter, and enclose the parameter name in pipes and exclamation points. For example, |!ParamDate!|. See Syntax in Requirements.
3. Select the **Save** icon.

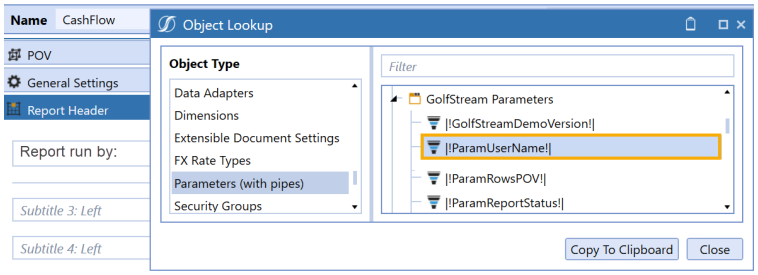
See these sample use cases:

- Assign to a Report Subheader
- Assign to a Cube View Column Header

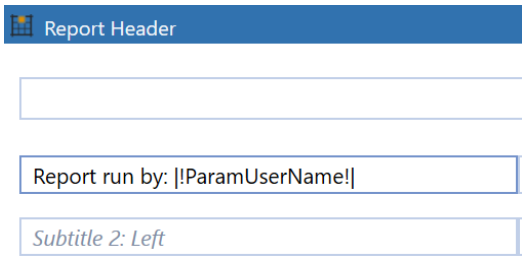
Assign to a Report Subheader

Perform these task to assign an Input Value parameter to prompt users to enter their name in a report header:

1. Select **Application** > **Presentation** > **Cube Views**.
2. Select **Cube View Group**, the appropriate Cube View, **Designer**, and then **Report Header**.
3. In **Subtitle,** enter **Report run by:**.
4. Select **Object Lookup**, and then **Parameters (with pipes)**.
5. Select the parameter to assign, and then select the **Copy to Clipboard** button.

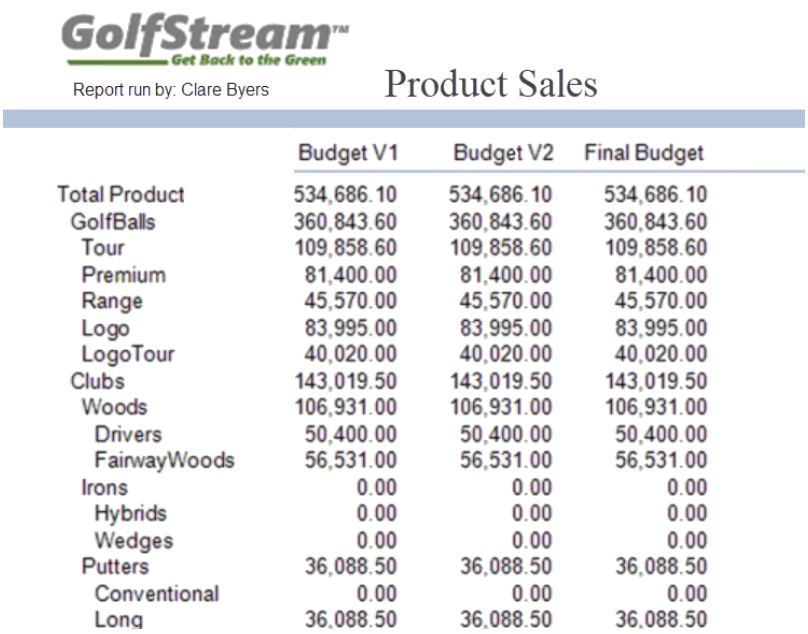


6. Paste the parameter after **Report run by:**, as shown:



7. Select **Save** and then **Open Data Explorer** to run the Cube View.

8. Enter your name and se;ect **Show Report**. The report generates, displaying your name as shown:



GolfStream™
Get Back to the Green

Report run by: Clare Byers

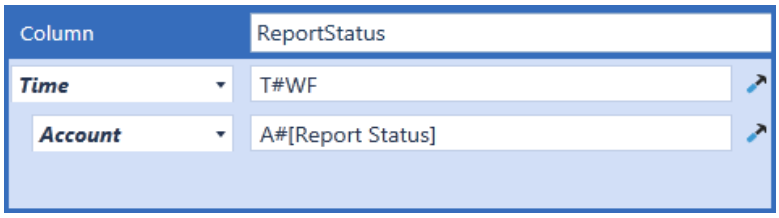
Product Sales

	Budget V1	Budget V2	Final Budget
Total Product	534,686.10	534,686.10	534,686.10
GolfBalls	360,843.60	360,843.60	360,843.60
Tour	109,858.60	109,858.60	109,858.60
Premium	81,400.00	81,400.00	81,400.00
Range	45,570.00	45,570.00	45,570.00
Logo	83,995.00	83,995.00	83,995.00
LogoTour	40,020.00	40,020.00	40,020.00
Clubs	143,019.50	143,019.50	143,019.50
Woods	106,931.00	106,931.00	106,931.00
Drivers	50,400.00	50,400.00	50,400.00
FairwayWoods	56,531.00	56,531.00	56,531.00
Irons	0.00	0.00	0.00
Hybrids	0.00	0.00	0.00
Wedges	0.00	0.00	0.00
Putters	36,088.50	36,088.50	36,088.50
Conventional	0.00	0.00	0.00
Long	36,088.50	36,088.50	36,088.50

Assign to a Cube View Column Header

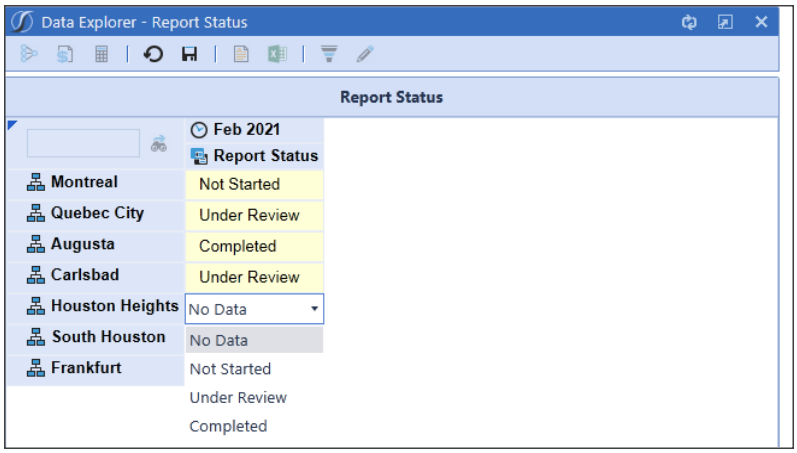
In this example, a Delimited List parameter is assigned to a sample GolfStream Cube View so users can indicate if reports are not started, under review, or completed for each entity.

1. Select **Application** > **Presentation** > **Cube Views**.
2. Select the appropriate Cube View.
3. Select **Rows and Columns** to see that column members display time based on workflow and a non-financial account called Report Status. The row members identify all base entities for Clubs.



4. Select a row header, and then **Data**.
5. In **Cell Type**, select **Combo Box**.
6. In **List Parameter**, enter **ParamReportStatus** and save the Cube View.

   When users run the Cube View, they can specify a report status for each entity, as shown:



Modify a Parameter

> **TIP:** Before editing a parameter, identify where it is used to understand the effects of modifying it. Select the parameter and select the  **Show Objects That Reference The Selected Item** icon.

1. Select **Application** > **Presentation** > **Workspaces**.
2. Select the dashboard maintenance unit that uses the parameter, and then **Parameters**.
3. Select the parameter to modify settings described in Create a Parameter.
4. Select **Save**.

Copy a Parameter

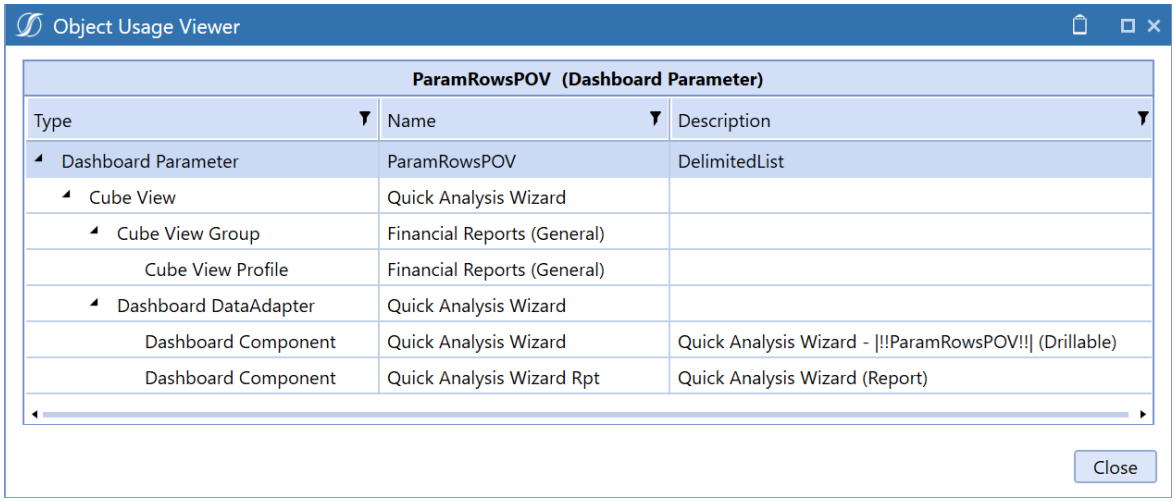
Copy a parameter to quickly create a new parameter that must function similarly.

1. Select **Application** > **Presentation** > **Workspaces**.
2. Select the dashboard maintenance unit in which the parameter is used, and then **Parameters**.
3. Select the parameter upon which to base a new parameter, then select **Copy**.
4. Select **Paste** and then **Rename** to specify a name for the new parameter.
5. Modify the new parameter as needed and assign it to an appropriate object. See:
    - Create a Parameter
    - Assign a Parameter

Delete or Remove a Parameter

You can delete a parameter after removing it from its assigned objects.

1. Select **Application** > **Presentation** > **Workspaces**.
2. Select the dashboard maintenance unit in which the parameter is used.
3. Select the parameter, then **Show Objects That Reference The Selected Item**. This identifies the objects from which you must remove the parameter. For example:

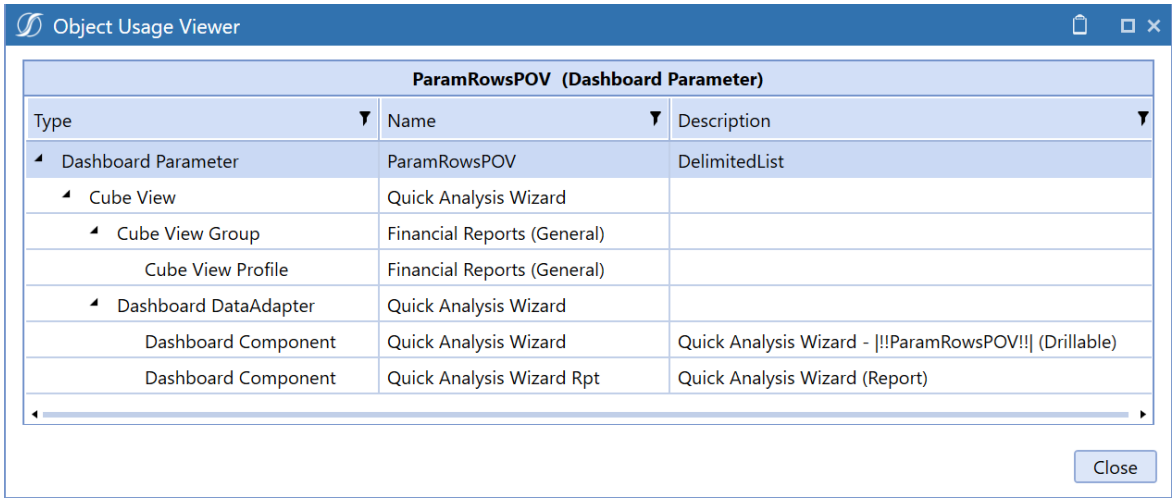


4. Select **Application** to find and edit the objects that use the parameter. For example, to remove a parameter from report header in a Cube View:
    a. Select **Cube View Groups**, the Cube View, and then **Designer**.
    b. Select **Report Header**, delete the parameter from the subtitle, and then select **Save**.
5. To delete the parameter:
    a. Repeat steps 2 - 4 to ensure you remove the parameter from all objects.
    b. Select the parameter and then select the **Delete** icon.

Rename a Parameter

> **CAUTION:** If you rename a parameter, you must re-assign it to all objects because the new name is not automatically applied. See Identify parameter assignments. Do not use special characters and dashes (-) in names.

1. Select **Application** > **Presentation** > **Workspaces**.
2. Selectthe dashboard maintenance unit that uses the parameter, and then **Parameters**.
3. Select the parameter, then **Show Objects That Reference The Selected Item** to identify the objects using the parameter. You must assign the renamed parameter to these objects. For example:

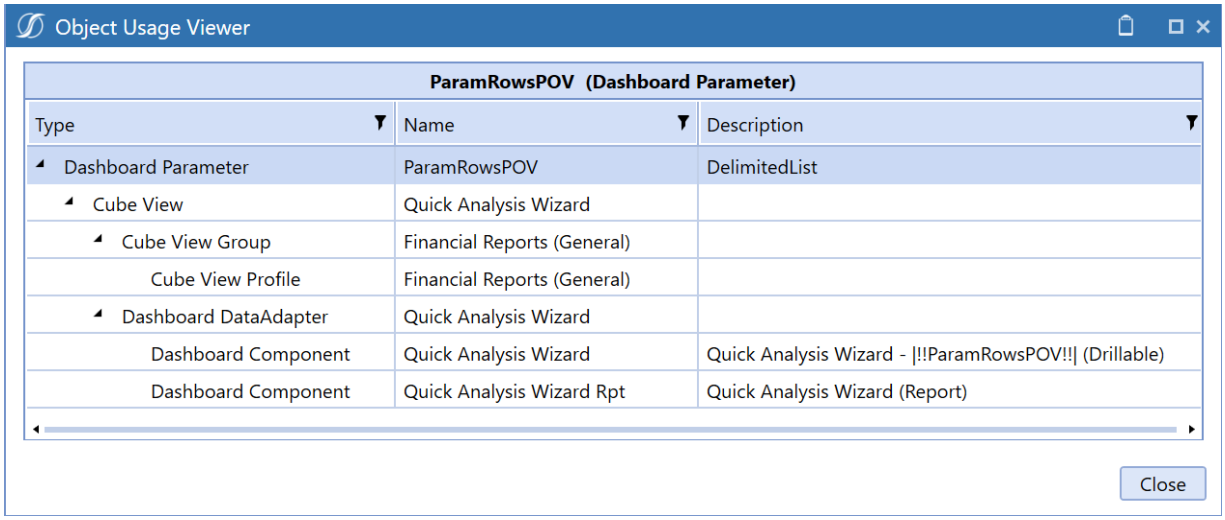


4. Select **Close**.
5. Select the parameter, then **Rename** and enter a different name.
6. Select **Save**.

Identify Parameter Assignments

Perform these steps to access a list of objects to which a parameter is assigned:

1. Select **Application** > **Presentation** > **Workspaces**.
2. Select the dashboard maintenance unit associated with the parameter, and then **Parameters**.
3. Select the parameter and then **Show Objects That Reference The Selected Item**. This lists all the objects that use the parameter.



Nest Parameters

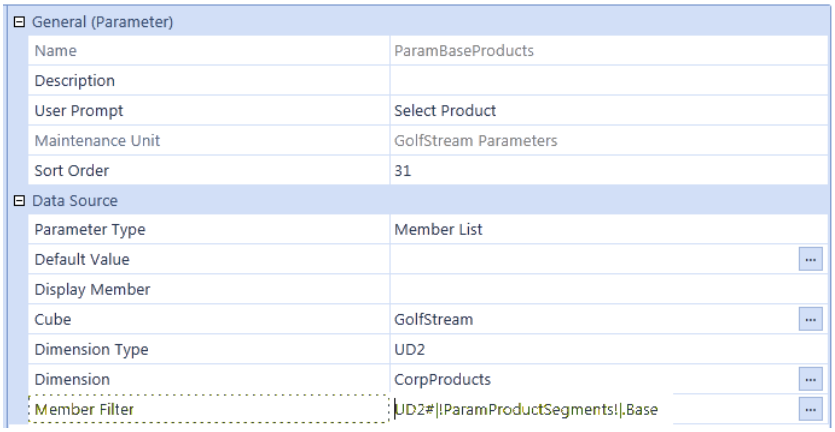
A nested parameter refers to another parameter, so you can further refine data selections for more granular data entry, reporting, and analysis. The following procedure based on the GolfStream application, describes how to create a nested parameter for a Cube View to display profit by product. The parameters defined below let users filter data by product segment, then by a specific product in a selected segment.

1. Select **Application** > **Presentation** > **Workspaces**.
2. Select the dashboard maintenance unit in which to store the parameters.
3. Select **Parameters**, and then **Create Parameter**.
4. Define the first parameter as shown below to prompt users to select a product segment.

⊟ General (Parameter)	
Name	ParamProductSegments
Description	
User Prompt	Select Product Segment
Maintenance Unit	GolfStream Selection Parameters
Sort Order	310
⊟ Data Source	
Parameter Type	Member List
Default Value	Clubs ...
Display Member	
Cube	GolfStream ...
Dimension Type	UD2 (SalesRegion)
Dimension	CorpProducts ...
Member Filter	UD2#Top.Children ...

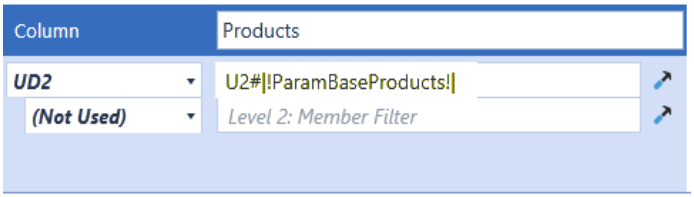
The Member Filter returns the child members of Top, in the UD2 dimension, in a drop-down menuso users can select a product segment. For example, if Top has 2 children, TotalProducts and TotalServices, users can select only those members.

5. Define the second parameter as shown below, referencing the first parameter in Member Filter.



The nested parameter will list the selected product segment's base members, so users can select a base-level member in TotalProducts orTotalServices.

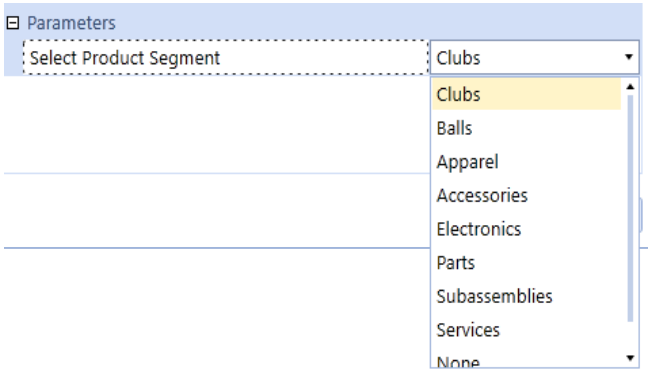
6. Save both parameters, and then select **Application** > **Presentation** > **Cube Views**.
7. Select the Cube View where you want to use the parameters.
8. Select the column or row that requires a parameter and enter the parameter name using pipes and exclamation points. See Syntax. The sample Cube View below displays products in columns, so a column is selected.



9. Select a dimension and select **Member Filter Builder** to ensure the ParamBaseProducts parameter references the ParamProductSegments parameter first, then returns a product list based on a selected segment.

10. Select **Open Data Explorer** to see how the nested parameters work.

    ParamProductSegements runs first, prompting you to select a segment.



    Then ParamBaseProducts runs, prompting you to select a product.

